

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

**PHÁT TRIỂN ỨNG DỤNG XÁC THỰC KHUÔN MẶT
SỬ DỤNG MẠNG NEURAL TÍCH CHẬP**

Sinh viên thực hiện: **Vũ Minh Thịnh**
MSSV: **63131330**
Giảng viên hướng dẫn: **TS. Nguyễn Đình Hưng**
Khoa: **Công Nghệ Thông Tin**

Khánh Hòa, tháng 6/2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

**PHÁT TRIỂN ỨNG DỤNG XÁC THỰC KHUÔN MẶT
SỬ DỤNG MẠNG NEURAL TÍCH CHẬP**

Sinh viên thực hiện: **Vũ Minh Thịnh**
MSSV: **63131330**
Giảng viên hướng dẫn: **TS. Nguyễn Đình Hưng**
Khoa: **Công Nghệ Thông Tin**

Khánh Hòa, tháng 6/2025



NGUỒN CHÍNH

1	Submitted to Nha Trang University Bài của Học sinh	8%
2	Submitted to Thai Nguyen University of Education Bài của Học sinh	4%
3	www.ctu.edu.vn Nguồn Internet	2%
4	Submitted to National Economics University Bài của Học sinh	1%
5	Submitted to The Olympia School Bài của Học sinh	1%
6	Ton Duc Thang University Xuất bản	1%
7	tailieu.vn Nguồn Internet	1%
8	Submitted to Vietnam Maritime University Bài của Học sinh	<1%
9	lib.uet.vnu.edu.vn Nguồn Internet	<1%
10	Nguyen Ba Duy, To Thanh Hai, Dinh Thanh Nhan, Truong Quoc Dinh. "FACIAL RECOGNITION FOR SUPERVISION OF ONLINE CLASSROOM", FAIR KỶ YẾU HỘI NGHỊ KHOA HỌC CÔNG NGHỆ QUỐC GIA LẦN THỨ XVII NGHIÊN CỨU CƠ BẢN VÀ ỨNG DỤNG CÔNG NGHỆ THÔNG TIN - Proceedings of the 17th	<1%

National Conference on Fundamental and
Applied Information Technology Research
(FAIR'2024), 2024

Xuất bản

11	Submitted to Ho Chi Minh City Open University	<1 %
	Bài của Học sinh	
12	Hanoi National University of Education	<1 %
	Xuất bản	
13	Phenikaa University	<1 %
	Xuất bản	
14	traichocunglulu.com	<1 %
	Nguồn Internet	
15	portal.ptit.edu.vn	<1 %
	Nguồn Internet	
16	Hanoi Pedagogical University 2	<1 %
	Xuất bản	
17	jst.dntu.edu.vn	<1 %
	Nguồn Internet	
18	text.123docz.net	<1 %
	Nguồn Internet	
19	Submitted to Foreign Trade University	<1 %
	Bài của Học sinh	
20	Submitted to Hanoi Metropolitan University	<1 %
	Bài của Học sinh	
21	fpt.ai	<1 %
	Nguồn Internet	
22	Submitted to University of West London	<1 %
	Bài của Học sinh	
23	fdocuments.net	<1 %
	Nguồn Internet	

24	laptrinhx.com Nguồn Internet	<1 %
25	anchor.fm Nguồn Internet	<1 %
26	vdocuments.net Nguồn Internet	<1 %
27	vbee.vn Nguồn Internet	<1 %
28	gust.edu.vn Nguồn Internet	<1 %
29	Submitted to University of Economics & Law Bài của Học sinh	<1 %
30	vi.d2l.ai Nguồn Internet	<1 %
31	123docz.net Nguồn Internet	<1 %
32	infonet.vietnamnet.vn Nguồn Internet	<1 %
33	www.slideserve.com Nguồn Internet	<1 %
34	Do Hoang Tu, Tran Van Lang, Pham Cong Xuyen, Le Mau Long. "PREDICTING THE ACTIVITY OF CHEMICAL COMPOUNDS BASED ON MACHINE LEARNING APPROACHES", FAIR KỶ YẾU HỘI NGHỊ KHOA HỌC CÔNG NGHỆ QUỐC GIA LẦN THỨ XVI Nghiên cứu cơ bản và Ứng dụng công nghệ thông tin Proceedings of the 16th National Conference on Fundamental and Applied Information Technology Research (FAIR'2023), 2023 Xuất bản	<1 %

Vietnam National University of Forestry

35	Xuất bản	<1 %
36	Submitted to Ho Chi Minh University of Technology and Education Bài của Học sinh	<1 %
37	sdh.duytan.edu.vn Nguồn Internet	<1 %
38	www.fit.hcmus.edu.vn Nguồn Internet	<1 %
39	huggingface.co Nguồn Internet	<1 %
40	top10app.net Nguồn Internet	<1 %
41	Submitted to Da Nang University of Economics Bài của Học sinh	<1 %
42	admin.ump.edu.vn Nguồn Internet	<1 %
43	www.keepandshare.com Nguồn Internet	<1 %
44	Submitted to Industrial University of Ho Chi Minh City Bài của Học sinh	<1 %
45	Trần Văn Quang, Trần Đình Toàn, Lê Mậu Long, Lê Minh Hưng. "PHÂN LOẠI HẠCH UNG THƯ PHỔI BẰNG MÔ HÌNH 3D-CNN", KỶ YẾU HỘI NGHỊ KHOA HỌC CÔNG NGHỆ QUỐC GIA LẦN THỨ XIII NGHIÊN CỨU CƠ BẢN VÀ ỨNG DỤNG CÔNG NGHỆ THÔNG TIN - Proceedings of the 13th National Conference on Fundamental & Applied Information Technology Research, 2020 Xuất bản	<1 %

46	techinsight.com.vn Nguồn Internet	<1 %
47	Submitted to Van Lang University Bài của Học sinh	<1 %
48	www.vietnamworks.com Nguồn Internet	<1 %

Loại trừ Trích dẫn Mở
Loại trừ mục lục tham khảo

Loại trừ trùng khớp < 20 words

TRƯỜNG ĐẠI HỌC NHA TRANG

Khoa/Viện: Công nghệ thông tin

PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ ĐÒ ÁN / KHÓA LUẬN TỐT NGHIỆP*(Dùng cho CBHD và nộp cùng báo cáo ĐA/KLTN của sinh viên)*

Tên đề tài: Phát Triển Úng Dụng Xác Thực Khuôn Mặt Sử Dụng Mạng Neural Tích Chập

Chuyên ngành: Công nghệ thông tin

Họ và tên sinh viên: Vũ Minh Thịnh. Mã sinh viên: 63131330 Lớp: 63.CNTT-CLC

Người hướng dẫn (học hàm, học vị, họ và tên): TS. Nguyễn Đình Hưng

Cơ quan công tác: Khoa Công nghệ thông tin - Trường Đại học Nha Trang

Phản đánh giá và cho điểm của người hướng dẫn (tính theo thang điểm 10)

Tiêu chí đánh giá	Trọng số (%)	Mô tả mức chất lượng				Điểm
		Giỏi	Khá	Đạt yêu cầu	Không đạt	
		9 - 10	7 - 8	5 - 6	< 5	
Xây dựng đề cương nghiên cứu	10					
Tinh thần và thái độ làm việc	10					
Kiến thức và kỹ năng làm việc	10					
Nội dung và kết quả đạt được	40					
Kỹ năng viết và trình bày báo cáo	30					
ĐIỂM TỔNG						

Ghi chú: Điểm tổng làm tròn đến 1 số lẻ.

Nhận xét chung (sau khi sinh viên hoàn thành ĐA/KLTN):

.....
.....
.....
.....

.Đồng ý cho sinh viên: Được bảo vệ: Không được bảo vệ:

Khánh Hòa, ngày.....tháng.....năm

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC NHA TRANG**Khoa/Viện:** Công nghệ thông tin**PHIẾU CHẤM ĐIỂM ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP***(Dành cho cán bộ chấm phản biện)*

Tên đề tài: Phát Triển Ứng Dụng Xác Thực Thực Khuôn Mặt Sử Dụng Mạng Neural Tích Chập

Chuyên ngành: Công Nghệ Thông Tin - PHOE

Họ và tên sinh viên: Vũ Minh Thịnh.

Mã sinh viên: 63131330

Người phản biện (học hàm, học vị, họ và tên):

.....

Cơ quan công tác:

I. Phần đánh giá và cho điểm của người phản biện (tính theo thang điểm 10)

Tiêu chí đánh giá	Trọng số (%)	Mô tả mức chất lượng				Điểm
		Giỏi	Khá	Đạt yêu cầu	Không đạt	
		9 - 10	7 - 8	5 - 6	< 5	
Hình thức bản thuyết minh	20					
Nội dung bản thuyết minh	30					
Mức độ trích dẫn và sao chép	20					
Kết quả nghiên cứu	30					
ĐIỂM TỔNG						

*Ghi chú: Điểm tổng làm tròn đến 1 số lẻ.***Kết luận:**

Đồng ý cho sinh viên:

Được bảo vệ: Không được bảo vệ: *Khánh Hòa, ngày.....tháng.....năm.....***Cán bộ chấm phản biện***(Ký và ghi rõ họ tên)*

LỜI CAM ĐOAN

Tôi, Vũ Minh Thịnh, xin cam kết rằng toàn bộ nội dung nghiên cứu với đề tài “**Phát triển ứng dụng xác thực khuôn mặt sử dụng mạng neural tích chập**” là kết quả của quá trình học tập, nghiên cứu nghiêm túc và nỗ lực cá nhân của tôi, được thực hiện dưới sự hướng dẫn tận tâm của Thầy Nguyễn Đình Hưng.

Tôi đảm bảo rằng các kết quả được trình bày trong nghiên cứu này là trung thực, không sao chép hoặc lấy từ bất kỳ nguồn nào mà không được cho phép.

Tôi khẳng định rằng nghiên cứu này được thực hiện một cách minh bạch, tuân thủ đầy đủ các quy định về học thuật và đạo đức nghiên cứu của Khoa Công nghệ thông tin và Trường Đại học Nha Trang. Tất cả các tài liệu tham khảo sử dụng trong quá trình thực hiện đều được trích dẫn đúng quy định và đầy đủ thông tin nguồn.

Nếu có bất kỳ phát hiện nào liên quan đến hành vi sao chép hoặc gian lận học thuật, tôi xin hoàn toàn chịu trách nhiệm và chấp nhận mọi hình thức xử lý từ phía Khoa và Nhà trường theo đúng quy định.

Tôi xin gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Đình Hưng vì đã luôn hỗ trợ và định hướng cho tôi trong suốt quá trình thực hiện đề tài này.

Khánh Hòa, ngày tháng năm 2025

Sinh viên thực hiện

Vũ Minh Thịnh

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành và sâu sắc đến quý thầy cô Khoa Công nghệ thông tin, Trường Đại học Nha Trang, những người đã luôn tận tình giảng dạy, hỗ trợ và tạo điều kiện thuận lợi cho em trong suốt quá trình học tập và nghiên cứu.

Em đặc biệt biết ơn Thầy Nguyễn Đình Hưng – người đã trực tiếp hướng dẫn, cung cấp tài liệu và hỗ trợ em hết mình trong suốt quá trình thực hiện đề tài “*Phát triển ứng dụng xác thực khuôn mặt sử dụng mạng neural tích chập*”. Những định hướng chuyên môn, sự động viên và giúp đỡ của Thầy là yếu tố quan trọng giúp em hoàn thiện đề tài một cách hiệu quả và chuyên nghiệp hơn.

Em cũng xin trân trọng cảm ơn Ban Giám hiệu, quý thầy cô trong Khoa và toàn thể Trường Đại học Nha Trang đã tạo ra một môi trường học tập và nghiên cứu năng động, hiện đại – nơi em có thể phát triển bản thân và theo đuổi các ý tưởng học thuật của mình.

Dù đã nỗ lực hết mình trong quá trình thực hiện đề tài, song với giới hạn về kiến thức và kinh nghiệm, em không thể tránh khỏi những thiếu sót. Em rất mong nhận được những góp ý quý báu từ quý thầy cô để có thể hoàn thiện hơn nữa kỹ năng và tư duy nghiên cứu trong tương lai.

Em xin chân thành cảm ơn!

MỤC LỤC

LỜI CAM ĐOAN	ix
LỜI CẢM ƠN	x
DANH MỤC HÌNH ẢNH	xiv
DANH MỤC BẢNG BIỂU	xvi
DANH MỤC TỪ VIẾT TẮT	xvi
Chương 1. TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU	1
1.1 LÝ DO CHỌN ĐỀ TÀI.....	1
1.2 MỤC TIÊU NGHIÊN CỨU	1
1.3 PHẠM VI NGHIÊN CỨU.....	2
1.4 NỘI DUNG NGHIÊN CỨU	2
1.5 PHƯƠNG PHÁP NGHIÊN CỨU VÀ CÁCH TIẾP CẬN	2
1.5.1 Cách tiếp cận	2
1.5.2 Phương pháp nghiên cứu	2
1.6 CÁC NGHIÊN CỨU LIÊN QUAN	2
1.6.1 Tình hình nghiên cứu trên thế giới	2
1.6.2 Tình hình nghiên cứu trong nước	3
Chương 2. CƠ SỞ LÍ THUYẾT	4
2.1 TỔNG QUAN VỀ HỌC SÂU	4
2.1.1 Học sâu (Deep Learning).....	4
2.1.2 Cách hoạt động của DeepLearning	5
2.1.3 Ưu điểm và nhược điểm của Học sâu	6
2.1.4 Một số loại hệ thống học sâu.....	7
2.2 THUẬT TOÁN HAAR CASCADE (OPENCV)	8
2.2.1 Thuật toán Haar Cascade	8
2.2.2 Đặc trưng Haar (Haar-like features)	9
2.2.3 Huấn luyện AdaBoost.....	10

2.2.4 Phân loại theo tầng (Cascade Classifier)	10
2.3 MẠNG NEURAL TÍCH CHẬP (Convolutional Neural Network).....	11
2.3.1 CNN (Convolutional Neural Network)	11
2.3.2 Cấu trúc mạng Convolutional Neural Network.....	18
2.3.3 Ứng dụng của Convolutional Neural Network	18
2.3.4 Các ứng dụng chính của Convolutional Neural Network.....	19
2.3.5 Một số mô hình CNN phổ biến trong bài toán nhận diện khuôn mặt....	21
2.4 FACENET	25
2.4.1 Nguồn gốc FaceNet	25
2.4.2 Các khái niệm cơ bản	26
2.5 GOOGLE ML KIT FACE DETECTION	28
2.6 CÔNG CỤ ĐƯỢC SỬ DỤNG.....	29
2.6.1 Python:.....	30
2.6.2 OpenCV trong Python.....	31
2.6.3 Google Colab	33
2.6.4 Jetpack Compose	35
2.7 LFW dataset (Labeled Faces in the Wild)	38
2.7.1 Nguồn gốc và mục đích.....	39
2.7.2 Cấu trúc và nội dung	39
2.7.3 Ưu điểm và hạn chế	39
2.7.4 Ứng dụng trong nghiên cứu	40
2.7.5 Tầm quan trọng và triển vọng	40
2.8 CelebA DATASET (CelebFaces Attributes Dataset).....	41
2.8.1 Nguồn gốc và mục đích.....	41
2.8.2 Cấu trúc và nội dung	42
2.8.3 Ưu điểm và hạn chế	42
2.8.4 Ứng dụng trong nghiên cứu	43

Chương 3. XÂY DỰNG CÔNG CỤ ĐIỂM DANH BẰNG NHẬN DIỆN KHUÔN MẶT	44
3.1 PHÂN TÍCH BÀI TOÁN	44
3.1.1 Mô tả bài toán	44
3.2 XÂY DỰNG MÔ HÌNH CHO NHẬN DẠNG Khuôn mặt	44
3.2.1 Tiền xử lý ảnh.....	44
3.2.2 Xuất kết quả ra JSON.....	46
3.3 XÂY DỰNG MÔ HÌNH	47
3.3.1 Tăng cường dữ liệu đã được tiền xử lý.....	47
3.3.2 Kiến trúc mô hình.....	49
3.3.3 Huấn luyện và đánh giá mô hình	49
3.3.4 Chuyển đổi sang TfLite (TensorFlow Lite).....	54
3.3.5 So sánh giữa file H5 và TFLite	55
3.4 XÂY DỰNG ỨNG DỤNG	56
3.4.1 Mô tả ứng dụng	56
3.4.2 Thành phần chính của ứng dụng	57
3.5 QUY TRÌNH SỬ DỤNG CÔNG CỤ.....	60
3.5.1 Quy trình xử lý dữ liệu.....	60
3.5.2 Quy trình xây dựng mô hình	60
3.5.3 Quy trình chuyển đổi.....	61
3.5.4 Tích hợp trên thiết bị di động.....	62
3.5.5 Sử dụng trên thiết bị di động	63
Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	65
4.1 Kết luận	65
4.2 Hướng phát triển.....	65
Chương 5. TÀI LIỆU THAM KHẢO.....	67

DANH MỤC HÌNH ẢNH

Số hiệu	Tên hình	Trang
Hình 2.1	Mối quan hệ giữa AI, ML và DL.	5
Hình 2.2	Quy trình xây dựng một giải pháp Học sâu cơ bản	7
Hình 2.3	Đặc trưng Haar	10
Hình 2.4	Áp dụng đặc trưng Haar trong nhận diện khuôn mặt	11
Hình 2.5	Nguyên lý hoạt động của Cascade Classifier	12
Hình 2.6	Một ví dụ khác về Average Pooling	15
Hình 2.7	Hàm ReLU	17
Hình 2.8	Cấu trúc mạng CNN	22
Hình 2.9	Phân loại (Classification)	23
Hình 2.10	Định vị (Localization)	23
Hình 2.11	Phát hiện (Detection)	24
Hình 2.12	Phân đoạn (Segmentation)	24
Hình 2.13	Nguyên lí hoạt động của FaceNet	25
Hình 2.14	Nguyên lí hoạt động của OpenFace	27
Hình 2.15	Mô hình ResNet	28
Hình 2.16	Kiến trúc của LeNet-5	29
Hình 2.17	Nhận diện khuôn mặt bằng FaceNet	30
Hình 2.18	Vector Embedding	31
Hình 2.19	Cách hoạt động của hàm Triplet Loss	32
Hình 2.20	Nhận diện và xác định các đặc điểm trên khuôn mặt bằng Google ML Kit	34
Hình 2.21	Nhận diện khuôn mặt bằng OpenCV	39
Hình 2.22	Một ví dụ về Jetpack Compose	44
Hình 2.23	Bộ dữ liệu Labeled Faces in the Wild (LFW)	45
Hình 2.24	Bộ dữ liệu CelebFaces Attributes (CelebA)	48
Hình 3.1	Ảnh ban đầu	53
Hình 3.2	Ảnh sau khi tiền xử lý	53
Hình 3.3	Ảnh ban đầu (2)	53
Hình 3.4	Ảnh sau khi tiền xử lý (2)	53

Hình 3.5	Data Augmentation	57
Hình 3.6	Biểu đồ loss của lần huấn luyện đầu tiên	64
Hình 3.7	Biểu đồ loss của lần huấn luyện thứ 2	64
Hình 3.8	Biểu đồ loss của lần huấn luyện thứ 3	64
Hình 3.9	Biểu đồ loss của lần huấn luyện thứ 4	64
Hình 3.10	Biểu đồ loss của lần huấn luyện thứ 5.	60
Hình 3.11	Điểm F1 (F1-score)	60
Hình 3.12	Ma trận nhầm lẫn (Confusion matrix)	62
Hình 3.13	Giao diện điểm danh của sinh viên	72
Hình 3.14	Giao diện xuất file CSV	76

DANH MỤC BẢNG BIỂU

Số hiệu	Tên bảng	Trang
Bảng 3.1	Sự khác nhau giữa H5 và TFLite	64-65

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa đầy đủ
CNN	Convolutional Neural Network (Mạng nơ-ron tích chập)
CSV	Comma-Separated Values (Giá trị phân cách bằng dấu phẩy)
GAN	Generative Adversarial Network (Mạng nơ-ron tạo sinh)
GPU	Graphics Processing Unit (Bộ xử lý đồ họa)
GRU	Gated Recurrent Unit (Đơn vị hồi quy có cổng)
JSON	JavaScript Object Notation (Ký hiệu đối tượng JavaScript)
LFW	Labeled Faces in the Wild (Khuôn mặt được gắn nhãn trong môi trường tự nhiên)
LSTM	Long Short-Term Memory (Bộ nhớ ngắn hạn dài)
ML	Machine Learning (Học máy)
MSSV	Mã số sinh viên
MTCNN	Multi-task Convolutional Neural Network (Mạng nơ-ron tích chập đa nhiệm)
ReLU	Rectified Linear Unit (Đơn vị tuyến tính chỉnh lưu)
RNN	Recurrent Neural Network (Mạng nơ-ron hồi quy)
SGD	Stochastic Gradient Descent (Hạ gradient ngẫu nhiên)
TFLite	TensorFlow Lite

Chương 1. TỔNG QUAN VỀ VĂN ĐỀ NGHIÊN CỨU

1.1 LÍ DO CHỌN ĐỀ TÀI

Hiện tại, quy trình điểm danh sinh viên tại các lớp học của Trường Đại học Nha Trang chủ yếu được thực hiện theo phương pháp thủ công, chẳng hạn như gọi tên hoặc ghi chép trên danh sách giấy. Phương pháp này đòi hỏi giảng viên phải dành nhiều thời gian và công sức, đặc biệt trong các lớp học có số lượng sinh viên đông đúc. Việc thực hiện thủ công không chỉ gây mất thời gian mà còn tiềm ẩn nhiều nguy cơ sai sót, như nhầm lẫn khi ghi nhận hoặc bỏ sót thông tin, đặc biệt khi sinh viên vắng mặt hoặc có sự thay đổi thường xuyên trong lớp học.

Bên cạnh đó, công nghệ nhận diện khuôn mặt đã đạt được nhiều tiến bộ đáng kể trong những năm gần đây, nhờ sự hỗ trợ của các công cụ tiên tiến như Google ML Kit và FaceNet (TensorFlow Lite). Tuy nhiên, việc áp dụng công nghệ này trên các thiết bị di động vẫn đối mặt với không ít thách thức, đặc biệt trong điều kiện ánh sáng yếu, góc quay không phù hợp hoặc sự đa dạng về biểu cảm khuôn mặt. Những hạn chế này đặt ra yêu cầu cần nghiên cứu và cải tiến để đảm bảo độ chính xác và hiệu quả trong thực tế.

Với mục tiêu giảm bớt gánh nặng công việc cho giảng viên, em quyết định lựa chọn đề tài xây dựng một ứng dụng nhận diện khuôn mặt phục vụ điểm danh sinh viên. Giải pháp này không chỉ giúp tự động hóa quy trình điểm danh mà còn mang lại sự chính xác và minh bạch trong quản lý danh sách sinh viên, từ đó tiết kiệm thời gian và công sức cho hoạt động giảng dạy hàng ngày.

Bằng cách triển khai đề tài, em hy vọng sẽ mang lại lợi ích thiết thực, không chỉ đáp ứng yêu cầu tốt nghiệp mà còn hỗ trợ cải thiện quy trình quản lý giáo dục tại trường. Đây là động lực lớn để em quyết tâm thực hiện đề tài với sự đầu tư nghiêm túc và hiệu quả.

1.2 MỤC TIÊU NGHIÊN CỨU

Mục tiêu chính của đề tài là xây dựng một ứng dụng di động dựa trên nền tảng Android, sử dụng công nghệ nhận diện khuôn mặt để tự động hóa quy trình điểm danh sinh viên tại Trường Đại học Nha Trang. Ứng dụng này nhằm thay thế phương pháp thủ công hiện tại, giúp tiết kiệm thời gian và giảm thiểu sai sót trong quản lý danh sách sinh viên, đặc biệt trong các lớp học đông đúc.

1.3 PHẠM VI NGHIÊN CỨU

Phạm vi nghiên cứu tập trung vào thiết kế một mô hình nhận diện khuôn mặt từ bộ dữ liệu có sẵn, phục vụ điểm danh sinh viên tại Trường Đại học Nha Trang với quy mô lớp học từ nhỏ đến lớn. Tuy nhiên, phạm vi giới hạn trong nhận diện sinh viên đã đăng ký, không hỗ trợ điều kiện ánh sáng yếu hoặc góc quay phức tạp, và thử nghiệm chỉ thực hiện tại một số lớp học cụ thể của trường.

1.4 NỘI DUNG NGHIÊN CỨU

- Thu thập và tiền xử lý bộ dữ liệu hình ảnh khuôn mặt sinh viên
- Nghiên cứu tập trung vào việc thiết kế và tối ưu hóa mô hình
- Kiểm thử và đánh giá mô hình.
- Triển khai và thử nghiệm trên thiết bị di động.

1.5 PHƯƠNG PHÁP NGHIÊN CỨU VÀ CÁCH TIẾP CẬN

1.5.1 Cách tiếp cận

- Thu thập, tiền xử lí dữ liệu.
- Nghiên cứu và lựa chọn mô hình học sâu tối ưu nhất.
- Huấn luyện, thử nghiệm và đánh giá kết quả.
- Áp dụng, triển khai thử nghiệm trên thiết bị.

1.5.2 Phương pháp nghiên cứu

- Áp dụng học máy và xử lý ảnh, tiền xử lí dữ liệu đầu vào.
- Sử dụng các thư viện có sẵn để xây dựng mô hình nhận diện khuôn mặt từ bộ dữ liệu đã được xử lí.
- Thử nghiệm, chỉnh sửa để tối ưu mô hình.

1.6 CÁC NGHIÊN CỨU LIÊN QUAN

1.6.1 Tình hình nghiên cứu trên thế giới

Trên thế giới, công nghệ nhận diện khuôn mặt đã được nghiên cứu và ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm giáo dục và quản lý điểm danh. Các nghiên cứu tập trung vào việc cải thiện độ chính xác và hiệu suất trên các thiết bị di động. Năm 2015, nhóm nghiên cứu tại Google đã phát triển FaceNet, một mô hình sử dụng CNN để trích xuất đặc trưng khuôn mặt, đạt độ chính xác cao trên bộ dữ liệu LFW với tỷ lệ lỗi dưới 1% (Nguồn: [FaceNet: A Unified Embedding for Face Recognition and Clustering](#)). Mô hình này đã được ứng dụng trong các hệ thống điểm danh tự động tại nhiều trường học ở Mỹ. Ngoài ra, các nghiên cứu gần đây từ Đại học Stanford (2023) đã cải tiến FaceNet bằng cách tích hợp kỹ thuật học sâu như Transformer,

giảm thời gian xử lý trên thiết bị di động xuống dưới 0,5 giây mỗi khuôn mặt, phù hợp với môi trường lớp học đồng (Nguồn: [FaceXFormer: A Unified Transformer for Facial Analysis](#)).

1.6.2 Tình hình nghiên cứu trong nước

Trong nước, công nghệ nhận diện khuôn mặt đã bắt đầu được nghiên cứu và ứng dụng trong giáo dục, tuy nhiên vẫn còn hạn chế về quy mô và độ chính xác. Các nghiên cứu chủ yếu tập trung vào việc thử nghiệm các mô hình có sẵn. Năm 2022, nhóm nghiên cứu tại Đại học Bách Khoa Hà Nội đã phát triển một ứng dụng điểm danh sử dụng nhận diện khuôn mặt dựa trên mô hình MTCNN (Multi-task Convolution Neutral Network). Ứng dụng này đạt độ chính xác khoảng 85% trong điều kiện ánh sáng tốt, nhưng hiệu suất giảm đáng kể khi ánh sáng yếu, cho thấy hạn chế trong môi trường thực tế (Nguồn: [Xây Dựng Nhận Diện Khuôn Mặt Bằng MTCNN Và FaceNet](#)). Một nghiên cứu tại Đại học FPT vào năm 2024 đã cải tiến bằng cách tích hợp mô hình FaceNet vào ứng dụng Android. Kết quả cho thấy độ chính xác tăng lên 90% trong môi trường lớp học, nhưng tốc độ xử lý vẫn chưa được tối ưu trên các thiết bị có cấu hình thấp (Nguồn: [Sinh viên FPT Edu hiện kế điểm danh tự động bằng nhận diện khuôn mặt](#)).

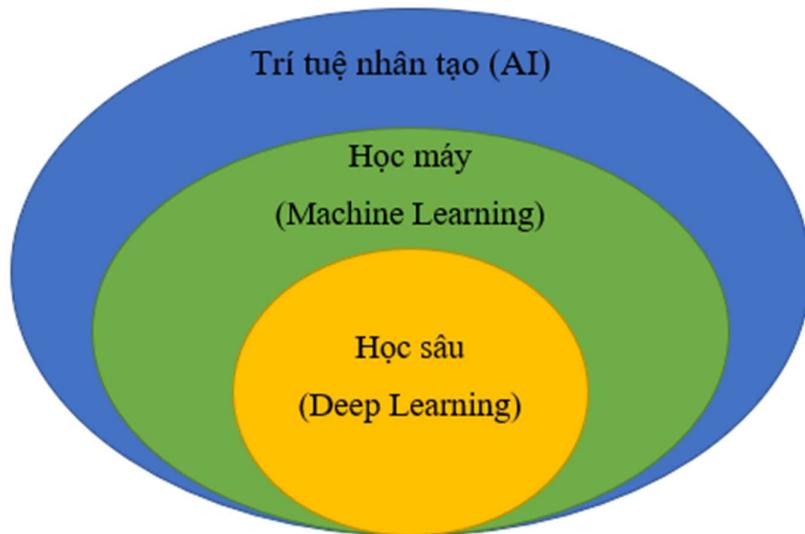
Chương 2. CƠ SỞ LÍ THUYẾT

2.1 TỔNG QUAN VỀ HỌC SÂU

Học sâu (Deep Learning) là một nhánh quan trọng của học máy (Machine Learning), tập trung vào việc sử dụng các mạng nơ-ron nhân tạo (Artificial Neural Networks) với nhiều tầng ẩn (hidden layers) để học và trích xuất các đặc trưng từ dữ liệu. Khác biệt với các phương pháp học máy truyền thống, học sâu nổi bật nhờ khả năng tự động học các biểu diễn dữ liệu ở nhiều cấp độ trừu tượng khác nhau. Nhờ đó, nó có thể mô hình hóa các mối quan hệ phức tạp trong dữ liệu mà không cần đến sự can thiệp thủ công trong việc trích xuất đặc trưng. Chính điều này đã giúp học sâu đạt được những thành tựu ấn tượng trong nhiều lĩnh vực như xử lý ảnh, nhận diện giọng nói và xử lý ngôn ngữ tự nhiên.

2.1.1 Học sâu (Deep Learning)

Học sâu là một phương pháp học máy sử dụng các mạng nơ-ron sâu (deep neural networks) để giải quyết các bài toán phức tạp. Các mạng nơ-ron sâu bao gồm nhiều tầng xử lý, trong đó mỗi tầng có khả năng học các đặc trưng từ dữ liệu đầu vào và truyền thông tin đến các tầng tiếp theo. Điểm nổi bật của học sâu là khả năng tự động trích xuất các đặc trưng quan trọng từ dữ liệu thô – ví dụ như hình ảnh hoặc âm thanh – mà không cần phải xác định trước các đặc trưng cụ thể, giúp giảm thiểu sự phụ thuộc vào kỹ thuật thủ công.



Hình 2.1: Mối quan hệ giữa AI, ML và DL.

2.1.2 Cách hoạt động của DeepLearning

Học sâu (Deep Learning) hoạt động bằng cách sử dụng các mạng nơ-ron nhân tạo với nhiều tầng ẩn để học các đặc trưng từ dữ liệu thô thông qua quá trình huấn luyện. Mỗi tầng trong mạng nơ-ron sẽ xử lý dữ liệu đầu vào và truyền thông tin đến tầng tiếp theo, giúp mô hình tự động trích xuất các đặc trưng phức tạp.

Quá trình này bắt đầu với việc đưa dữ liệu đầu vào, chẳng hạn như hình ảnh khuôn mặt,... Các tầng đầu tiên thường phát hiện các đặc trưng đơn giản như cạnh hoặc đường nét, trong khi các tầng sâu hơn kết hợp những đặc trưng này để nhận diện các đối tượng phức tạp hơn, như đặc điểm khuôn mặt cụ thể, thông qua các phép toán như tích chập và gộp.

Để tối ưu hóa mô hình, Deep Learning sử dụng thuật toán lan truyền ngược (backpropagation) kết hợp với tối ưu hóa như gradient descent. Thuật toán này điều chỉnh trọng số của mạng dựa trên sai số giữa đầu ra dự đoán và giá trị thực tế, giúp mô hình học tốt hơn qua từng lượt huấn luyện, đặc biệt hiệu quả với dữ liệu lớn như bộ dữ liệu của đề tài. Sau khi huấn luyện, mô hình có thể tạo ra các vector embedding hoặc phân loại trực tiếp. Kết quả phụ thuộc vào chất lượng dữ liệu và cách điều chỉnh các siêu tham số như tốc độ học và số lượng tầng. Mỗi nơ-ron có một hàm kích hoạt, có nhiệm vụ chuẩn hóa đầu ra từ nơ-ron đó. Khi dữ liệu được đưa vào mạng nơ-ron, nó sẽ đi qua tất cả các lớp và trả về kết quả tại lớp cuối cùng, gọi là lớp đầu ra (output layer).

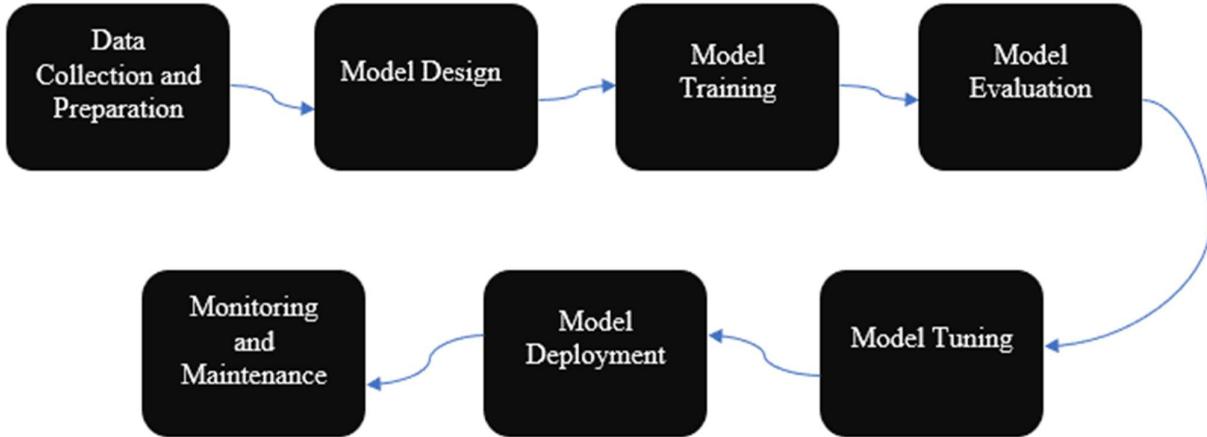
Trong quá trình huấn luyện mô hình mạng nơ-ron, các trọng số này sẽ được điều chỉnh. Mục tiêu của mô hình là tìm ra bộ giá trị trọng số tối ưu để đưa ra các dự đoán chính xác nhất. Các hệ thống học sâu yêu cầu phần cứng mạnh mẽ để xử lý lượng dữ liệu lớn và thực hiện các phép tính phức tạp. Nhiều mô hình học sâu có thể mất hàng tuần, thậm chí hàng tháng để huấn luyện trên các phần cứng tiên tiến nhất hiện nay.

Vậy, học sâu (deep learning) khác gì so với học máy (machine learning)?

Học sâu (Deep Learning) và học máy (Machine Learning) đều là các nhánh của trí tuệ nhân tạo, tuy nhiên học máy sử dụng các thuật toán để học từ dữ liệu và đưa ra dự đoán, thường yêu cầu trích xuất đặc trưng thủ công. Trong khi học sâu sử dụng mạng nơ-ron sâu với nhiều tầng ẩn để tự động trích xuất đặc trưng từ dữ liệu thô, như hình ảnh hoặc âm thanh, mà không cần can thiệp thủ công. Học máy thường hiệu quả hơn trên tập dữ liệu nhỏ và đơn giản, trong khi học sâu cần dữ liệu lớn để đạt độ chính xác

cao. Học máy dễ triển khai và giải thích hơn, còn học sâu phức tạp hơn và thường được coi là “hộp đen” do khó hiểu cách mô hình đưa ra quyết định.

Điều này giúp học sâu vượt trội trong các bài toán phức tạp như nhận diện khuôn mặt, nhưng đòi hỏi dữ liệu lớn và tài nguyên tính toán mạnh mẽ hơn.



Hình 2.2 Quy trình xây dựng một giải pháp Học sâu cơ bản.

2.1.3 Ưu điểm và nhược điểm của Học sâu

2.1.3.1 Ưu điểm

Học sâu là một bước ngoặt lớn trong lĩnh vực trí tuệ nhân tạo, cho phép ta xây dựng các mô hình với độ chính xác rất cao trong nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên, giọng nói và nhiều lĩnh vực khác. Học sâu mang lại nhiều lợi thế vượt trội, cụ thể:

- Tự động trích xuất đặc trưng từ dữ liệu thô, như hình ảnh khuôn mặt, mà không cần can thiệp thủ công, giúp tiết kiệm thời gian và công sức so với học máy truyền thống.
- Đạt hiệu suất cao trong các bài toán phức tạp như nhận diện khuôn mặt, ví dụ FaceNet trong đề tài có độ chính xác cao khi xử lý dữ liệu lớn.
- Có khả năng tổng quát hóa tốt nếu được huấn luyện trên dữ liệu đa dạng, giúp ứng dụng nhận diện khuôn mặt hoạt động ổn định với nhiều sinh viên khác nhau.
- Linh hoạt trong việc mở rộng và ứng dụng vào nhiều lĩnh vực khác nhau, từ giáo dục đến an ninh, tạo tiềm năng phát triển lâu dài cho các hệ thống nhận diện.

- Hiệu năng tốt: Có khả năng thực hiện tính toán song song và xử lý lượng dữ liệu lớn.

2.1.3.2 Nhược điểm

Mặc dù có nhiều ưu điểm, Học sâu (Deep Learning) có một số hạn chế đáng chú ý như:

- Cần lượng dữ liệu lớn để hoạt động hiệu quả, nếu bộ dữ liệu không đủ, kết quả sẽ kém.
- Phụ thuộc vào tài nguyên tính toán mạnh như GPU, gây khó khăn khi chạy trên các thiết bị có cấu hình thấp.
- Quá trình huấn luyện mất nhiều thời gian, đặc biệt với mô hình như FaceNet, đòi hỏi kiên nhẫn trong phát triển.
- Khó hiểu cách mô hình ra quyết định, giống như một “hộp đen”, làm khó kiểm tra và sửa lỗi khi nhận diện sai.

2.1.4 Một số loại hệ thống học sâu

Học sâu (Deep Learning) là một lĩnh vực phát triển mạnh mẽ, bao gồm nhiều loại hệ thống khác nhau được thiết kế để giải quyết các bài toán đa dạng trong xử lý dữ liệu. Dưới đây là một số hệ thống học sâu phổ biến:

- Mạng nơ-ron tích chập (Convolutional Neural Networks - CNNs):** CNNs là một kiến trúc được phát triển đặc biệt để xử lý dữ liệu hình ảnh và video. CNN sử dụng các tầng tích chập (convolutional layers) để tự động trích xuất đặc trưng không gian, chẳng hạn như cạnh, kết cấu hoặc mẫu, thông qua các bộ lọc học được, kết hợp với các tầng gộp (pooling layers) để giảm kích thước dữ liệu mà vẫn giữ được thông tin quan trọng.
- Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNNs):** RNNs là một loại hệ thống học sâu được thiết kế để xử lý dữ liệu chuỗi thời gian hoặc tuần tự, như văn bản, giọng nói hoặc chuỗi số. RNN hoạt động bằng cách duy trì một trạng thái ẩn (hidden state) qua các bước thời gian, cho phép mô hình học các mối quan hệ phụ thuộc giữa các phần tử trong chuỗi. Tuy nhiên, loại mạng này thường gặp vấn đề về mất thông tin dài hạn khi chuỗi quá dài, ảnh hưởng đến hiệu suất.
- Mạng nơ-ron hồi quy dài hạn (Long Short-Term Memory - LSTM):** Để khắc phục nhược điểm của RNN, LSTM được phát triển như một biến thể nâng cấp. LSTM giới thiệu các cổng (gates) như cổng quên, cổng đầu vào và cổng đầu ra, giúp kiểm

soát luồng thông tin trong mạng, từ đó duy trì thông tin quan trọng qua nhiều bước thời gian. Loại mạng này thường được sử dụng trong các bài toán như dự đoán chuỗi thời gian, nhận diện giọng nói hoặc dịch máy nhờ khả năng ghi nhớ dài hạn.

- d) **Mạng no-ron Gated Recurrent Unit (GRU):** Một biến thể khác của RNN là GRU, được tối ưu hóa để giảm độ phức tạp so với LSTM bằng cách sử dụng ít công hơn, bao gồm công cập nhật và công đặt lại. GRU giữ được hiệu suất gần tương đương LSTM nhưng nhẹ hơn về mặt tính toán, làm cho nó trở thành lựa chọn phổ biến trong các ứng dụng yêu cầu hiệu quả trên thiết bị có tài nguyên hạn chế, chẳng hạn như phân tích văn bản hoặc xử lý chuỗi ngắn.
- e) **Mạng no-ron tự chú ý (Transformer):** Transformer là một bước tiến quan trọng trong học sâu, nổi bật với cơ chế tự chú ý (self-attention) thay vì phụ thuộc vào các kết nối tuần tự như RNN. Transformer được thiết kế ban đầu cho xử lý ngôn ngữ tự nhiên, chẳng hạn như dịch máy hoặc tạo văn bản, nhưng gần đây đã được mở rộng sang các lĩnh vực khác nhờ khả năng xử lý song song và học các mối quan hệ phức tạp giữa các phần tử dữ liệu, cải thiện đáng kể tốc độ và hiệu suất.
- f) **Mạng no-ron tạo sinh (Generative Adversarial Networks - GANs):** GANs là một loại hệ thống học sâu bao gồm hai mạng: mạng tạo (generator) và mạng phân biệt (discriminator), cạnh tranh với nhau để tạo ra dữ liệu mới giống với dữ liệu thật. GAN thường được áp dụng để sinh ảnh, video hoặc âm thanh, dựa trên nguyên lý học không giám sát (unsupervised learning), và đã cho thấy tiềm năng lớn trong việc tạo dữ liệu tổng hợp cho các bài toán học máy khác.

2.2 THUẬT TOÁN HAAR CASCADE (OPENCV)

2.2.1 Thuật toán Haar Cascade

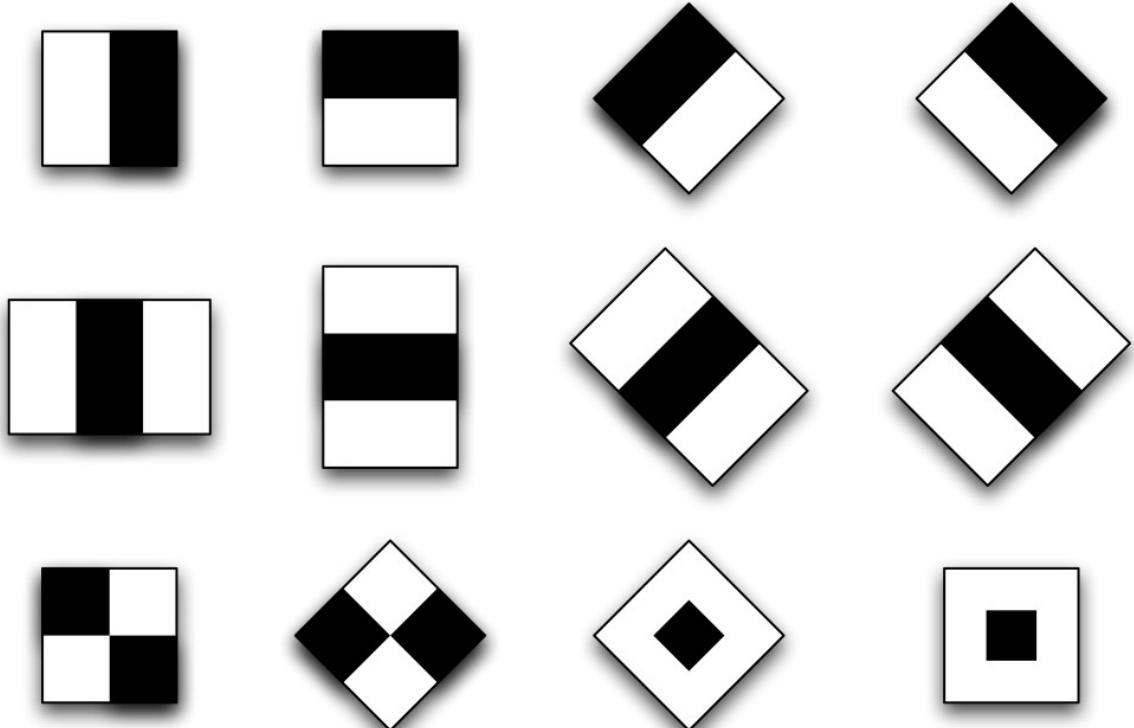
Thuật toán **Haar Cascade** (thuật toán phát hiện đối tượng dựa trên đặc trưng Haar và cascade classifier) được giới thiệu lần đầu năm 2001 bởi Paul Viola và Michael Jones trong bài báo “Rapid Object Detection using a Boosted Cascade of Simple Features“. Đây là một kỹ thuật học máy dùng để phát hiện các đối tượng (như khuôn mặt, mắt, biển số xe, v.v.) trong ảnh và video.

OpenCV cung cấp nhiều mô hình Haar Cascade đã huấn luyện sẵn (dưới dạng file XML) cho các đối tượng phổ biến như khuôn mặt, mắt, nụ cười, cơ thể..., lưu trong thư mục ‘haarcascades’ của thư viện OpenCV. Người dùng chỉ cần nạp bộ phân loại

(Classifier) tương ứng và gọi hàm ‘detectMultiScale’ để quét ảnh ở nhiều tỉ lệ và xác định vị trí các đối tượng.

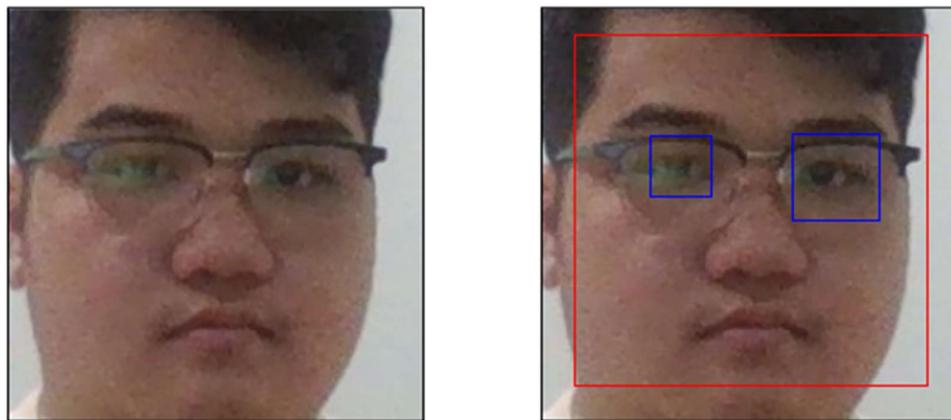
2.2.2 Đặc trưng Haar (Haar-like features)

Những đặc trưng Haar là các mẫu hình chữ nhật đơn giản dùng để mô tả vùng trên ảnh, được thiết kế dựa trên cơ sở hàm Haar. Mỗi đặc trưng gồm một hay nhiều hình chữ nhật phân vùng (màu đen và trắng), tính bằng hiệu tổng cường độ các pixel bên trong vùng trắng và vùng đen.



Hình 2.3 Đặc trưng Haar

Ví dụ, một đặc trưng Haar có thể đo sự khác biệt độ sáng giữa vùng mắt (thường tối) và vùng gò má (thường sáng) của khuôn mặt. Thiết kế này cho phép mô tả các tính chất cơ bản của khuôn mặt (như vùng mắt tối hơn má, sống mũi sáng hơn vùng mắt) chỉ với phép cộng và trừ đơn giản. Để tăng tốc việc tính toán tổng điểm ảnh trong các hình chữ nhật nêu trên, thuật toán sử dụng **anh tích phân** (summed-area table) – một cấu trúc tiền xử lý giúp tính tổng điểm ảnh bất kỳ trong vùng hình chữ nhật chỉ với bốn phép toán cộng/trừ, không phụ thuộc vào kích thước vùng. Nhờ vậy, các đặc trưng Haar có thể được tính rất nhanh trên nhiều vị trí và kích thước khác nhau của cửa sổ trượt (sliding window).



Hình 2.4 Áp dụng đặc trưng Haar trong nhận diện khuôn mặt

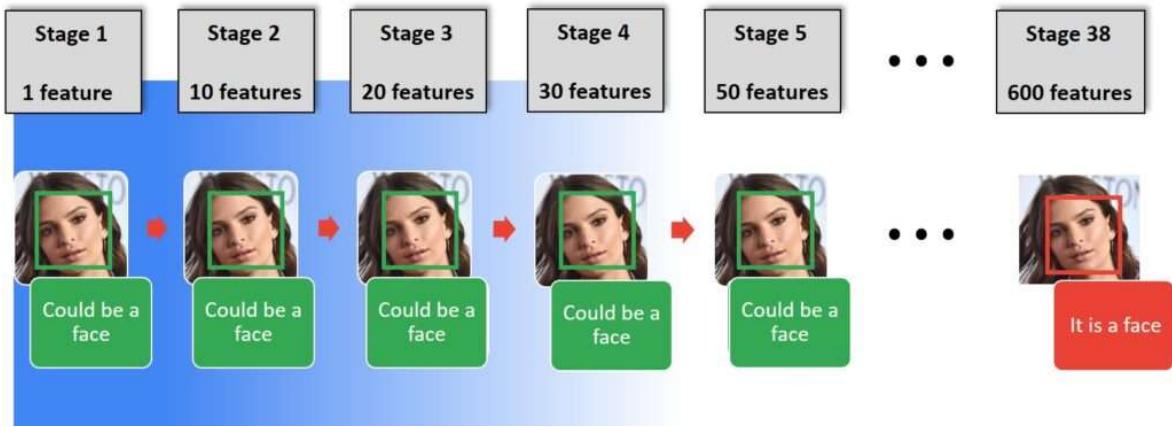
2.2.3 Huấn luyện AdaBoost

Thuật toán **AdaBoost** – một kỹ thuật tăng cường – trên bộ dữ liệu huấn luyện bao gồm nhiều ảnh có khuôn mặt và không có khuôn mặt. AdaBoost lặp đi lặp lại các bước cân nhắc trọng số ảnh, lựa chọn đặc trưng và ngưỡng phân tách tốt nhất để phân biệt mặt/nền, từ đó sinh ra một tập hợp các bộ phân loại “yếu” (weak classifier). Cuối cùng, các bộ phân loại yếu này được kết hợp thành bộ phân loại “mạnh” (strong classifier) theo tổ hợp tuyến tính có trọng số. Phương pháp này cho phép giảm từ hàng trăm nghìn đặc trưng xuống chỉ còn khoảng vài nghìn đặc trưng hiệu quả (thuật toán gốc còn chia thành hàng chục giai đoạn; trung bình chỉ khoảng 10 đặc trưng được đánh giá trên mỗi cửa sổ ảnh)

2.2.4 Phân loại theo tầng (Cascade Classifier)

Để xử lý ảnh nhanh hơn, ta có thể sử dụng cơ chế *cascade* – một chuỗi các giai đoạn phân loại liên tiếp. Mỗi giai đoạn của cascade là một bộ phân loại Haar kết hợp từ một tập con các đặc trưng đã chọn. Khi quét một cửa sổ con trên ảnh, cascade sẽ kiểm tra tuần tự các giai đoạn: nếu cửa sổ này không qua giai đoạn đầu tiên (nghĩa là không giống mặt), nó ngay lập tức bị loại và không cần xem xét thêm; nếu qua thì tiếp tục kiểm tra giai đoạn kế tiếp.

Cách làm này giúp loại nhanh các cửa sổ không phải mặt chỉ với vài phép tính đơn giản (các giai đoạn đầu thường dùng ít đặc trưng), chỉ tập trung xử lý sâu hơn những vùng khả nghi. Nếu một cửa sổ vượt qua tất cả các giai đoạn trong cascade, nó được coi là chứa đối tượng cần tìm (ví dụ khuôn mặt). Cơ chế này vừa đảm bảo độ nhạy tốt, vừa tăng tốc đáng kể nhờ bỏ qua các khu vực nhiễu.



Hình 2.5 Nguyên lý hoạt động của Cascade Classifier

Nguồn: <https://anuja-shukla.medium.com/face-mask-detection-using-opencv-d9a2d75e4064>

2.3 MẠNG NEURAL TÍCH CHẬP (CONVOLUTIONAL NEURAL NETWORK)

2.3.1 CNN (Convolutional Neural Network)

Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một loại mạng nơ-ron sâu được thiết kế chuyên biệt để xử lý dữ liệu có cấu trúc lưới, như hình ảnh hoặc chuỗi thời gian. CNN nổi bật nhờ khả năng tự động trích xuất đặc trưng không gian từ dữ liệu, rất hiệu quả trong các bài toán nhận diện và phân loại.

Ví dụ, với một ảnh kích thước 200x200 ở thang độ xám (1 kênh), lớp đầu vào có 40,000 nút, dẫn đến số lượng lớn trọng số (weights) và độ lệch (biases), làm phức tạp việc tính toán. Tuy nhiên, vì thông tin pixel thường phụ thuộc vào vùng lân cận, việc bỏ qua một số nút không ảnh hưởng nhiều đến độ chính xác. Cửa sổ tích chập được sử dụng để giảm số tham số, tập trung vào các đặc trưng cục bộ như đường viền hoặc vùng sáng tối.

Một ví dụ khác liên quan đến âm thanh là xử lý tín hiệu âm thanh dạng sóng (waveform) chuyển đổi thành ma trận 256x256 từ phổ Mel (Mel spectrogram) với 1 kênh, tương đương 65,536 nút đầu vào. Sử dụng mạng nơ-ron thông thường sẽ tạo ra số lượng tham số khổng lồ, gây khó khăn trong huấn luyện. Nhờ cửa sổ tích chập, mô hình có thể tập trung vào các đặc trưng như tần số hoặc nhịp điệu tại các khu vực nhỏ, giảm độ phức tạp mà vẫn giữ được thông tin quan trọng.

Cấu trúc của CNN bao gồm các tầng tích chập (convolutional layers) để trích xuất đặc trưng như cạnh hoặc mẫu, tầng gộp (pooling layers) để giảm kích thước dữ liệu, và tầng kết nối đầy đủ (fully connected layers) để đưa ra kết quả cuối cùng.

CNN hoạt động dựa trên các bộ lọc (filters) học được, quét qua dữ liệu đầu vào để phát hiện các đặc trưng cục bộ. Cơ chế này cho phép mô hình học các đặc trưng từ

đơn giản như đường nét đến phức tạp như hình dạng hoặc nhịp điệu, mà không cần trích xuất thủ công.

Một đặc điểm quan trọng của CNN là tính chất chia sẻ trọng số (weight sharing), giúp giảm số lượng tham số cần học, tăng hiệu quả tính toán và hạn chế nguy cơ overfitting khi xử lý dữ liệu lớn như hình ảnh hoặc âm thanh.

2.3.1.1 Lớp tích chập (Convolution Layer)

Lớp tích chập (Convolution Layer) là thành phần cốt lõi của mạng nơ-ron tích chập (CNN), chịu trách nhiệm trích xuất đặc trưng từ dữ liệu đầu vào như hình ảnh hoặc tín hiệu âm thanh. Lớp này sử dụng các bộ lọc (filters) hoặc nhân tử (kernels) để quét qua dữ liệu, phát hiện các mẫu cục bộ.

Quá trình tích chập diễn ra khi bộ lọc, thường có kích thước nhỏ như 3x3 hoặc 5x5, trượt qua từng vùng của dữ liệu đầu vào, thực hiện phép nhân tích chập để tạo ra các bản đồ đặc trưng (feature maps). Kết quả phụ thuộc vào trọng số của bộ lọc, được học trong quá trình huấn luyện.

Một đặc điểm quan trọng là lớp tích chập sử dụng tính chất chia sẻ trọng số, nghĩa là cùng một bộ lọc được áp dụng cho tất cả các vị trí trong dữ liệu, giúp giảm số lượng tham số và tăng hiệu quả tính toán, đặc biệt với dữ liệu lớn.

Ngoài ra, lớp tích chập thường kết hợp với hàm kích hoạt như ReLU (Rectified Linear Unit) để giới thiệu tính phi tuyến, giúp mô hình học được các mối quan hệ phức tạp hơn từ dữ liệu, nâng cao khả năng nhận diện các đặc trưng như cạnh hoặc kết cấu.

2.3.1.2 Lớp gộp (Pooling Layer)

Lớp gộp (Pooling Layer) là một thành phần quan trọng trong kiến trúc của mạng nơ-ron tích chập (Convolutional Neural Network – CNN). Lớp này thường được đặt sau lớp tích chập nhằm mục đích giảm số lượng đặc trưng, làm giảm chiều dữ liệu và giảm thiểu chi phí tính toán, đồng thời vẫn giữ lại được các đặc trưng quan trọng nhất của ảnh đầu vào.

Lớp gộp có vai trò:

- **Giảm chiều dữ liệu:** Bằng cách thu nhỏ kích thước của các bản đồ đặc trưng (feature map), lớp gộp giúp giảm số lượng tham số và phép tính trong mạng.
- **Tránh overfitting:** Việc rút gọn thông tin giúp mô hình tổng quát tốt hơn, ít bị lệ thuộc vào chi tiết nhỏ và nhiễu trong ảnh.

- **Tăng tính bất biến:** Lớp gộp giúp mạng trở nên bất biến (invariant) với các thay đổi nhỏ như dịch chuyển, xoay hoặc co giãn nhẹ trong ảnh đầu vào.

Hai loại lớp gộp phổ biến nhất trong thực tế là:

- **Max Pooling (Gộp giá trị lớn nhất):** Chọn giá trị lớn nhất trong từng vùng nhỏ (window) của bản đồ đặc trưng. Đây là phương pháp phổ biến nhất vì giúp giữ lại đặc trưng mạnh nhất trong mỗi vùng.

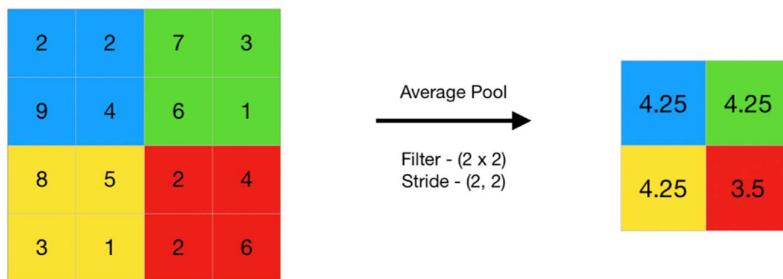
Ví dụ: Cho một cửa sổ 2×2 trượt trên ma trận:

1	3
2	4

Max pooling sẽ chọn giá trị lớn nhất là 4.

- **Average Pooling (Gộp trung bình):** Tính giá trị trung bình của tất cả các pixel trong mỗi vùng. Phương pháp này giữ lại mức thông tin tổng quát nhưng có thể làm mất đi các đặc trưng mạnh.

Ví dụ: Với cùng cửa sổ như trên, average pooling sẽ trả về $(1+3+2+4)/4 = 2.5$.



Hình 2.6 Một ví dụ khác về Average Pooling

Lớp gộp hoạt động bằng cách:

- Chọn một cửa sổ trượt (thường là 2×2 hoặc 3×3).
- Di chuyển cửa sổ đó qua bản đồ đặc trưng với bước trượt (stride) nhất định.
- Ứng dụng phép gộp lên từng vùng cửa sổ.

2.3.1.3 Hàm phi tuyến – ReLU

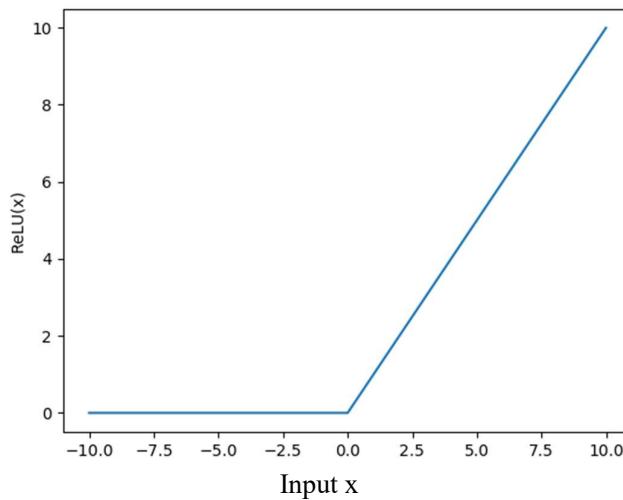
ReLU (Rectified Linear Unit) là một hàm phi tuyến đơn giản, có công thức toán học như sau: $f(x) = \max(0, x)$. Trong đó, x là đầu vào của hàm. Điều này có nghĩa là ReLU sẽ giữ nguyên các giá trị dương và đưa tất cả giá trị âm về 0.

Việc chèn hàm phi tuyến ReLU vào sau lớp tích chập mang lại nhiều lợi ích cho mạng nơ-ron, cụ thể:

- **Tăng tính phi tuyến:** Mặc dù lớp tích chập là tuyến tính, nhưng bài toán học đặc trưng trong thực tế thường mang tính phi tuyến. ReLU giúp mạng mô hình hóa được các mối quan hệ phi tuyến phức tạp trong dữ liệu.
- **Giảm hiện tượng gradient vanishing:** So với các hàm phi tuyến truyền thống như sigmoid hay tanh, ReLU giúp lan truyền gradient hiệu quả hơn trong quá trình huấn luyện, từ đó giúp mạng học nhanh hơn và giảm nguy cơ gradient bị tiêu biến.
- **Tính đơn giản và hiệu quả:** ReLU rất đơn giản về mặt tính toán (chỉ cần so sánh với 0), do đó có thể thực thi nhanh trên các phần cứng như GPU.
Ví dụ, giả sử đầu ra của lớp tích chập là: $[-3, -1, 0, 2, 5]$.
Khi áp dụng ReLU, ta thu được: $[0, 0, 0, 2, 5]$.

Dễ thấy rằng các giá trị âm bị loại bỏ, chỉ giữ lại các giá trị có khả năng đại diện cho các đặc trưng có ý nghĩa trong ảnh.

Tuy nhiên, ReLU cũng có nhược điểm là hiện tượng “dead neuron”, khi một số nơ-ron có thể bị vô hiệu hóa nếu giá trị đầu vào luôn âm, nhưng vấn đề này thường được khắc phục bằng các biến thể như Leaky ReLU hoặc Parametric ReLU.



Hình 2.7 Hàm ReLU

2.3.1.4 Lớp kết nối đầy đủ (Fully Connected Layer)

Lớp kết nối đầy đủ (Fully Connected Layer) là thành phần cuối cùng trong kiến trúc của mạng nơ-ron tích chập (CNN), được sử dụng để tổng hợp các đặc trưng đã trích xuất từ các tầng trước nhằm đưa ra kết quả cuối cùng.

Trong lớp này, tất cả các nơ-ron từ tầng trước (thường là sau lớp gộp hoặc lớp tích chập cuối) được kết nối với mọi nơ-ron trong tầng hiện tại, tạo ra một vector đặc trưng tổng quát. Vector này sau đó được sử dụng để phân loại hoặc dự đoán, chẳng hạn như xác định danh tính của một đối tượng.

Lớp kết nối đầy đủ thường đi kèm với hàm kích hoạt như softmax để tạo ra xác suất cho các lớp đầu ra, giúp mô hình phân loại hiệu quả hơn. Tuy nhiên, lớp này thường chứa nhiều tham số, dẫn đến nguy cơ overfitting nếu không được kiểm soát tốt.

Để giảm số lượng tham số và tăng hiệu quả, các kiến trúc hiện đại đôi khi thay thế lớp kết nối đầy đủ bằng các tầng tích chập toàn cục (global average pooling), nhưng lớp kết nối đầy đủ vẫn đóng vai trò quan trọng trong nhiều mô hình CNN truyền thống.

2.3.1.5 Softmax

Hàm Softmax là một hàm kích hoạt quan trọng trong mạng nơ-ron tích chập (CNN), thường được áp dụng ở lớp kết nối đầy đủ để chuyển đổi đầu ra thành các xác suất cho các lớp phân loại. Hàm này hoạt động bằng cách áp dụng hàm mũ (exponential) cho từng giá trị đầu vào, sau đó chuẩn hóa tổng các giá trị này về 1, đảm bảo tổng xác suất của tất cả các lớp bằng 100%.

Công thức cơ bản là

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Trong đó:

- + z_i là đầu vào của lớp Softmax tại vị trí i .
- + K là tổng số lớp (class).

Một lợi ích lớn của Softmax là cung cấp cách diễn giải trực quan, cho phép so sánh mức độ tự tin của mô hình đối với từng lớp, rất hữu ích trong các ứng dụng như nhận diện hoặc phân loại.

Tuy nhiên, hàm này có thể gặp vấn đề khi dữ liệu đầu vào có sự chênh lệch lớn, dẫn đến gradient quá lớn hoặc quá nhỏ, gây khó khăn trong huấn luyện. Để khắc phục, người ta thường chuẩn hóa dữ liệu trước khi áp dụng Softmax, đảm bảo mô hình hoạt động ổn định và hiệu quả hơn.

2.3.1.6 Adam

Thuật toán Adam (Adaptive Moment Estimation) là một phương pháp tối ưu hóa hiệu quả, thường được sử dụng trong quá trình huấn luyện mạng nơ-ron tích chập (CNN)

để cập nhật trọng số. Thuật toán này kết hợp lợi ích của Momentum và RMSProp, sử dụng cả gradient trung bình (momentum) và bình phương trung bình của gradient để điều chỉnh tốc độ học (learning rate) một cách thích ứng.

Adam hoạt động bằng cách duy trì hai giá trị trung bình: trung bình động của gradient (moment estimate) và trung bình động của bình phương gradient (RMS estimate). Điều này giúp mô hình hội tụ nhanh hơn và ổn định hơn so với các phương pháp như Gradient Descent thông thường, đặc biệt với các mạng phức tạp.

Một ưu điểm nổi bật của Adam là khả năng xử lý các bài toán có dữ liệu không đồng nhất hoặc nhiễu, nhưng nó có thể gặp vấn đề về hội tụ không ổn định nếu các siêu tham số (như beta1, beta2) không được điều chỉnh tốt, đòi hỏi sự cẩn trọng trong cài đặt.

Ngoài Adam, còn có một số phương pháp tối ưu hóa khác như:

- Gradient Descent (GD) là phương pháp cơ bản, cập nhật trọng số dựa trên gradient của hàm mất mát theo toàn bộ tập dữ liệu, nhưng thường chậm và không hiệu quả với dữ liệu lớn.
- Stochastic Gradient Descent (SGD) cải tiến từ GD bằng cách cập nhật trọng số dựa trên từng mẫu dữ liệu (hoặc một lô nhỏ), giúp tăng tốc độ huấn luyện. Ví dụ, khi huấn luyện CNN với tập dữ liệu 10,000 ảnh khuôn mặt, SGD có thể giảm thời gian huấn luyện từ 10 giờ (với GD) xuống còn 2 giờ, nhưng dễ bị dao động do tính ngẫu nhiên.
- Momentum là một biến thể của SGD, bổ sung quán tính để tăng tốc độ hội tụ, giúp vượt qua các điểm tối ưu cục bộ. Phương pháp này đặc biệt hiệu quả trong các bài toán có hàm mất mát phức tạp, nhưng cần điều chỉnh tham số quán tính cẩn thận.
- RMSProp (Root Mean Square Propagation) điều chỉnh tốc độ học dựa trên trung bình động của bình phương gradient, giúp ổn định hơn SGD trong các bài toán có gradient dao động mạnh, chẳng hạn như khi huấn luyện CNN trên dữ liệu không đồng nhất như ảnh khuôn mặt với ánh sáng khác nhau.
- Adamax: Một biến thể của Adam, có hiệu suất tốt hơn trên các mạng có độ dốc lớn.

Mỗi phương pháp tối ưu hóa có ưu điểm và hạn chế riêng, và hiệu suất của chúng có thể thay đổi tùy thuộc vào cấu trúc của mạng, dữ liệu dùng để huấn luyện và yêu cầu của bài toán.

2.3.1.7 Hàm mất mát Categorical_crossentropy

Hàm mất mát Categorical_crossentropy là một hàm mất mát phổ biến trong mạng nơ-ron tích chập (CNN), được sử dụng trong các bài toán phân loại đa lớp, đặc biệt khi đầu ra sử dụng hàm Softmax. Hàm này đo lường mức độ khác biệt giữa phân phối xác suất dự đoán của mô hình và phân phối thực tế (ground truth).

Công thức của CCE như sau:

$$CCE(y, \hat{y}) = - \sum_{i=1}^M y_i \log(\hat{y}_i)$$

Trong đó:

- + y_i là giá trị thực tế (0 hoặc 1) của lớp i
- + \hat{y}_i là xác suất dự đoán cho lớp i.
- + M là số lớp (class).

Một lợi ích của Categorical_crossentropy là khả năng làm việc tốt với các bài toán phân loại đa lớp, đặc biệt khi kết hợp với Softmax, nhưng nó có thể nhạy cảm với dữ liệu không cân bằng, đòi hỏi các kỹ thuật như trọng số lớp để xử lý hiệu quả.

Ngoài Categorical_crossentropy, còn có nhiều hàm mất mát khác được sử dụng trong mạng nơ-ron tích chập (CNN) tùy thuộc vào bài toán. Dưới đây là một số hàm mất mát phổ biến:

- Binary Crossentropy được áp dụng cho bài toán phân loại nhị phân, đo lường sự khác biệt giữa xác suất dự đoán và nhãn thực (0 hoặc 1), ví dụ khi nhận diện khuôn mặt chỉ có hai trạng thái (có hoặc không phải sinh viên). Công thức cơ bản là:

$$L = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

- Mean Squared Error (MSE) là hàm mất mát phổ biến trong các bài toán hồi quy, tính toán bình phương chênh lệch giữa giá trị dự đoán và giá trị thực tế, phù hợp khi đầu ra là giá trị liên tục.
- Triplet Loss là hàm mất mát đặc biệt trong nhận diện khuôn mặt, tối ưu hóa khoảng cách giữa ba mẫu: anchor, positive (cùng lớp), và negative (khác lớp).
- Kullback-Leibler Divergence (KL Divergence) thường được dùng để so sánh hai phân phối xác suất, hữu ích trong các bài toán học không giám sát khi tinh

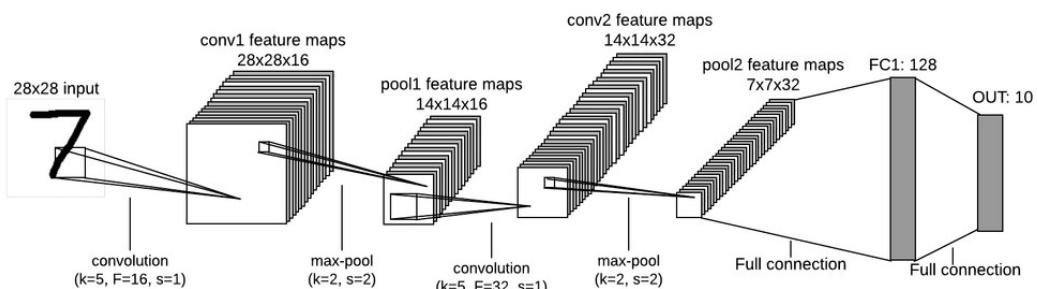
chỉnh mô hình, nhưng ít phô biến hơn trong nhận diện khuôn mặt. Các hàm này cung cấp sự linh hoạt khi thiết kế mô hình tùy theo yêu cầu cụ thể.

2.3.2 Cấu trúc mạng Convolutional Neural Network

Cấu trúc mạng Convolutional Neural Network (CNN) được thiết kế theo các tầng tuần tự nhằm trích xuất và phân loại đặc trưng từ dữ liệu không gian như hình ảnh. Mạng thường bắt đầu bằng các lớp tích chập (convolutional layers) để phát hiện các đặc trưng cục bộ.

Các lớp gộp (pooling layers) được thêm vào để giảm kích thước dữ liệu, giữ lại thông tin quan trọng và tăng khả năng bát biến. Ví dụ, trong một mạng CNN xử lý ảnh 224x224 pixel, lớp gộp 2x2 có thể giảm kích thước xuống còn 112x112, giúp giảm gánh nặng tính toán mà vẫn duy trì đặc trưng chính.

Sau các lớp tích chập và gộp, mạng thường bao gồm một hoặc nhiều lớp kết nối đầy đủ (fully connected layers) để tổng hợp thông tin và đưa ra dự đoán, thường kết hợp với hàm Softmax để phân loại. Số lượng và thứ tự các tầng có thể thay đổi tùy theo kiến trúc, như LeNet-5 hoặc ResNet.



Hình 2.8 Cấu trúc mạng CNN

Quá trình huấn luyện sử dụng các phương pháp tối ưu hóa như Adam và hàm mất mát như Categorical_crossentropy để điều chỉnh trọng số, đảm bảo mô hình học được các đặc trưng phức tạp từ dữ liệu, chẳng hạn như đặc điểm khuôn mặt trong nhận diện.

2.3.3 Ứng dụng của Convolutional Neural Network

Mạng nơ-ron tích chập (CNN) được ứng dụng rộng rãi trong nhiều lĩnh vực nhờ khả năng trích xuất đặc trưng không gian từ dữ liệu như hình ảnh và video. Một ứng dụng nổi bật là nhận diện khuôn mặt, như trong các hệ thống điểm danh sinh viên, nơi CNN giúp trích xuất đặc trưng và so sánh để xác định danh tính.

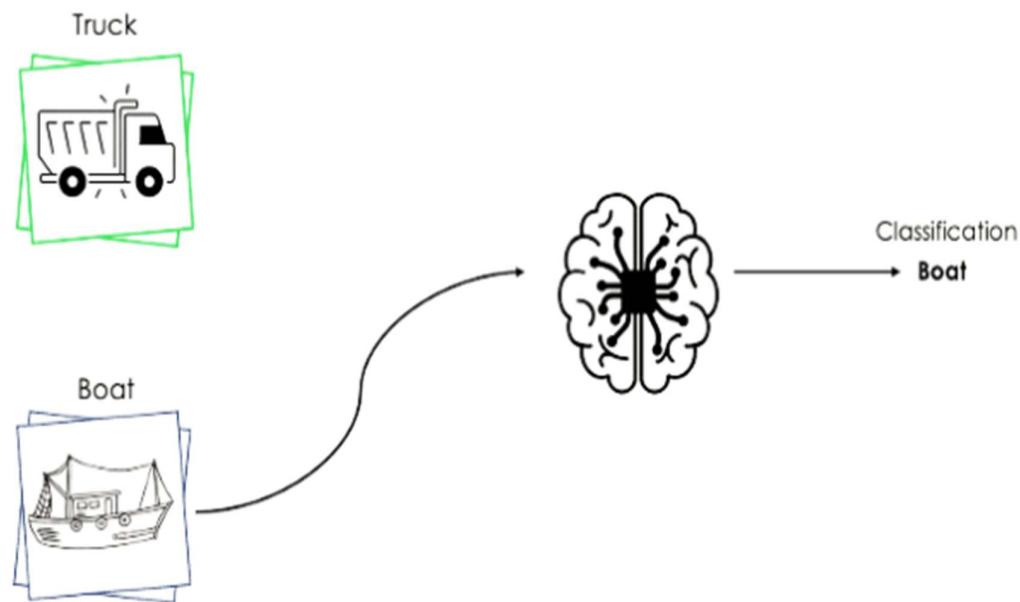
CNN cũng được sử dụng trong phân loại hình ảnh, ví dụ trong y tế, để phát hiện các bệnh lý qua ảnh X-quang hoặc MRI. Một mô hình CNN có thể phân loại ảnh X-

quang lồng ngực để xác định bệnh nhân có dấu hiệu viêm phổi với độ chính xác cao, hỗ trợ bác sĩ trong chẩn đoán.

Trong lĩnh vực xe tự hành, CNN đóng vai trò quan trọng trong nhận diện đối tượng trên đường, như phát hiện người đi bộ hoặc biển báo giao thông, giúp xe đưa ra quyết định an toàn. Ngoài ra, CNN còn được áp dụng trong xử lý ngôn ngữ tự nhiên (NLP), như phân loại văn bản hoặc nhận diện chữ viết tay, nhờ khả năng xử lý dữ liệu không gian hiệu quả.

2.3.4 Các ứng dụng chính của Convolutional Neural Network

Classification (Phân loại): Là nhiệm vụ phổ biến nhất, nhằm phân loại nội dung hình ảnh thành các danh mục (nhãn) như con xe tải, tàu thuyền hoặc các đối tượng khác, với kết quả là xác suất thuộc về từng lớp.



Hình 2.9 Phân loại (Classification)

Localization (Định vị): Tập trung vào xác định vị trí của đối tượng trong hình ảnh bằng cách tạo hộp giới hạn (bounding box), kết hợp với nhãn lớp. Ví dụ, trong hệ thống giám sát, định vị có thể xác định vị trí khuôn mặt sinh viên trên màn hình để hỗ trợ điểm danh.



Hình 2.10 Định vị (Localization)

Detection (Phát hiện): Mở rộng định vị bằng cách xác định tất cả các đối tượng trong ảnh với nhiều hộp giới hạn và nhãn lớp. Trong xe tự hành, CNN sử dụng phát hiện để nhận diện biển báo, xe cộ, và người đi bộ, đảm bảo an toàn khi lái xe, ví dụ phát hiện 5 đối tượng khác nhau trong một khung hình.



Hình 2.11 Phát hiện (Detection)

Segmentation (Phân đoạn): xác định cả nhãn lớp và đường viền chi tiết của mỗi đối tượng trong hình ảnh. Trong y tế, phân đoạn giúp đánh dấu khu vực khối u trong ảnh phổi, hỗ trợ phân tích sâu để chẩn đoán bệnh lý một cách chính xác.



Hình 2.12 Phân đoạn (Segmentation)

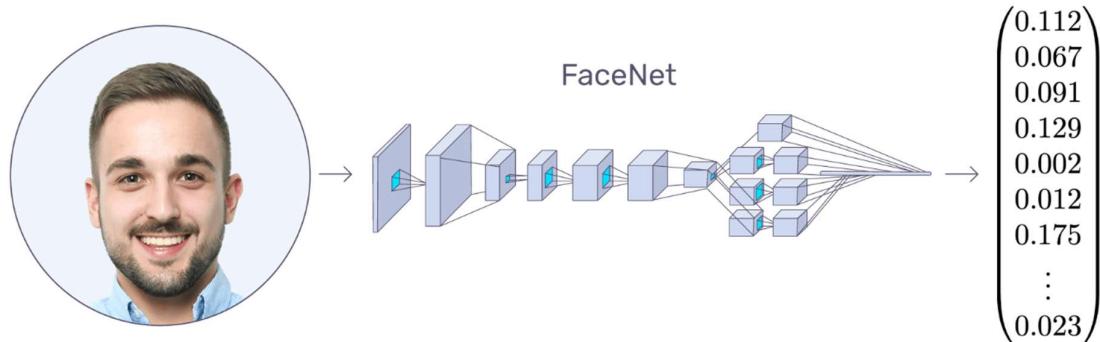
2.3.5 Một số mô hình CNN phổ biến trong bài toán nhận diện khuôn mặt

2.3.5.1 FaceNet

FaceNet là một mô hình CNN tiên tiến được phát triển bởi Google, được thiết kế đặc biệt cho các bài toán nhận diện và xác minh khuôn mặt với độ chính xác cao. Mô hình này sử dụng kiến trúc mạng sâu để ánh xạ hình ảnh khuôn mặt vào không gian embedding 128 chiều, tạo ra một biểu diễn số duy nhất cho mỗi khuôn mặt.

Điểm nổi bật của FaceNet là việc sử dụng hàm mất mát Triplet Loss, một kỹ thuật tối ưu hóa độc đáo nhằm giảm khoảng cách giữa các ảnh của cùng một người (anchor và positive) và tăng khoảng cách với ảnh của người khác (negative), từ đó đảm bảo tính phân biệt cao trong không gian vector. FaceNet không chỉ hiệu quả trong việc nhận diện khuôn mặt mà còn rất linh hoạt khi áp dụng vào các hệ thống xác thực danh tính hoặc giám sát an ninh, nhờ khả năng thích ứng với các điều kiện ánh sáng và góc quay khác nhau.

Tuy nhiên, hạn chế của FaceNet nằm ở yêu cầu dữ liệu huấn luyện lớn và cần được chuẩn hóa tốt để đạt hiệu suất tối ưu, đồng thời việc huấn luyện với Triplet Loss đòi hỏi kỹ thuật chọn mẫu phức tạp, làm tăng độ khó trong triển khai thực tế.



Hình 2.13 Nguyên lý hoạt động của FaceNet

Nguồn: <https://vnba.org.vn/vi/momo-ung-dung-deep-learning-trong-phat-hien-hanh-vi-gian-lan-2493.htm>

2.3.5.2 DeepFace

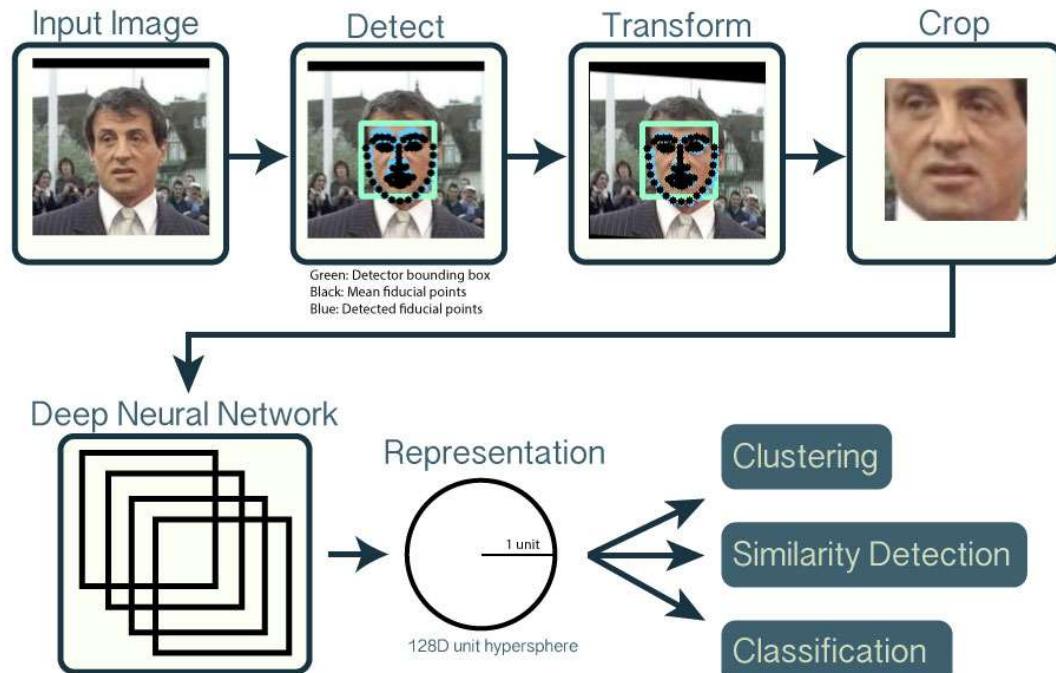
DeepFace là mô hình CNN được phát triển bởi Facebook, ra đời với mục tiêu mô phỏng cách con người nhận diện khuôn mặt, đạt độ chính xác ánh tượng gần 97% trên tập dữ liệu LFW (Labeled Faces in the Wild). Mô hình này sử dụng kiến trúc sâu với 9 tầng tích chập, kết hợp các kỹ thuật xử lý tiên tiến để trích xuất đặc trưng chi tiết từ hình ảnh khuôn mặt, bao gồm cả các đặc trưng từ các góc độ và điều kiện ánh sáng khác nhau.

DeepFace áp dụng kỹ thuật căn chỉnh khuôn mặt (face alignment) trước khi đưa vào mạng, giúp giảm ảnh hưởng của các yếu tố như xoay hoặc biến dạng, từ đó cải thiện hiệu suất nhận diện. Mô hình này phù hợp với các ứng dụng thực tế như quản lý danh tính trong mạng xã hội, giám sát an ninh, hoặc hệ thống điểm danh tự động nhờ khả năng xử lý dữ liệu phức tạp.

Tuy nhiên, DeepFace yêu cầu tài nguyên tính toán lớn để huấn luyện và triển khai, đặc biệt khi xử lý các tập dữ liệu lớn với nhiều biến thể. Ngoài ra, hiệu suất của mô hình có thể giảm khi gặp các ảnh có chất lượng thấp hoặc điều kiện ánh sáng không lý tưởng, đòi hỏi phải cải thiện dữ liệu đầu vào trước khi xử lý.

2.3.5.3 OpenFace

OpenFace là một mô hình CNN mã nguồn mở, được xây dựng dựa trên các ý tưởng từ DeepFace và FaceNet, nhằm cung cấp một giải pháp nhận diện khuôn mặt hiệu quả trên các thiết bị có tài nguyên hạn chế. Mô hình này tạo ra vector embedding 128 chiều để biểu diễn khuôn mặt và sử dụng các kỹ thuật như dLib để phát hiện khuôn mặt trước khi tiến hành phân tích, giúp giảm đáng kể thời gian xử lý và tối ưu hóa hiệu suất trên các thiết bị thông thường như điện thoại hoặc máy tính bảng.



Hình 2.14 Nguyên lý hoạt động của OpenFace

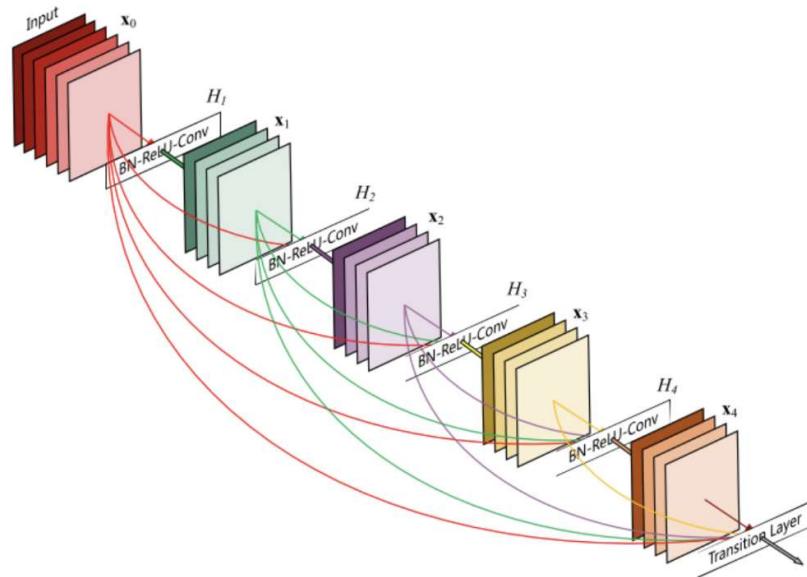
Nguồn: <https://cmusatyalab.github.io/openface/>

OpenFace được thiết kế với mục tiêu dễ dàng tích hợp vào các ứng dụng thực tế, chẳng hạn như các hệ thống điểm danh tự động, giám sát an ninh, hoặc nhận diện danh tính trong các ứng dụng thương mại. Một ưu điểm lớn của OpenFace là tính linh hoạt và khả năng tùy chỉnh, cho phép các nhà phát triển điều chỉnh mô hình theo nhu cầu cụ thể mà không cần đầu tư phần cứng cao cấp.

Tuy nhiên, do là mô hình mã nguồn mở, OpenFace có thể không đạt độ chính xác cao như FaceNet hoặc DeepFace trong các điều kiện phức tạp, đặc biệt khi xử lý dữ liệu với nhiều nhiễu hoặc biến dạng, và cần được cải tiến thêm để đáp ứng các yêu cầu khắt khe trong các hệ thống quy mô lớn.

2.3.5.4 ResNet (Residual Networks)

ResNet (Residual Networks) là một dòng mô hình CNN mang tính cách mạng, được phát triển để giải quyết các vấn đề trong huấn luyện mạng sâu, đặc biệt là hiện tượng gradient vanishing khi số tầng tăng lên đáng kể. Điểm nổi bật của ResNet là việc sử dụng các kết nối tắt (skip connections), cho phép gradient truyền trực tiếp qua các tầng mà không bị suy giảm, từ đó giúp mạng học sâu hiệu quả hơn.



Hình 2.15 Mô hình ResNet

Nguồn: <https://blog.csdn.net/shuiCSDN/article/details/104001092>

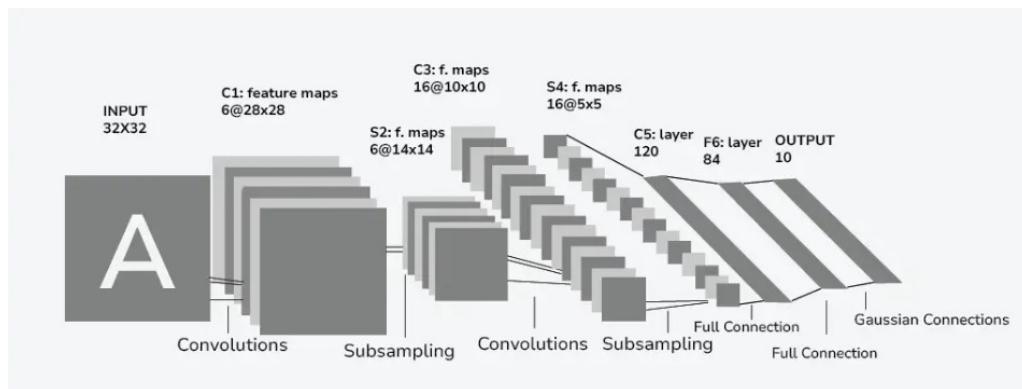
Các phiên bản phổ biến của ResNet, như ResNet-50, ResNet-101, và ResNet-152, được tối ưu hóa để đạt hiệu suất cao trên các tập dữ liệu lớn như ImageNet, đồng thời được ứng dụng rộng rãi trong các bài toán nhận diện khuôn mặt, phân loại hình ảnh, và phát hiện đối tượng. ResNet không chỉ cải thiện hiệu suất mà còn tăng tính ổn định

trong quá trình huấn luyện, cho phép xử lý các tập dữ liệu phức tạp với nhiều biến thể khác nhau, chẳng hạn như hình ảnh khuôn mặt với các góc độ và điều kiện ánh sáng đa dạng.

Tuy nhiên, nhược điểm của ResNet nằm ở yêu cầu tài nguyên tính toán lớn, đặc biệt với các phiên bản có nhiều tầng, làm hạn chế khả năng triển khai trên các thiết bị có phần cứng yếu. Ngoài ra, việc thiết kế và tối ưu hóa các kết nối tắt đòi hỏi sự điều chỉnh cẩn thận để đạt hiệu quả tốt nhất trong từng bài toán cụ thể.

2.3.5.5 LeNet-5

LeNet-5 là một trong những mô hình CNN đầu tiên, được Yann LeCun phát triển vào những năm 1990, đánh dấu bước ngoặt quan trọng trong việc ứng dụng mạng nơ-ron tích chập vào các bài toán thị giác máy tính. Mô hình này được thiết kế chủ yếu để nhận diện chữ số viết tay trong tập dữ liệu MNIST, với cấu trúc đơn giản gồm 5 tầng: 2 tầng tích chập, 2 tầng gộp, và 1 tầng kết nối đầy đủ. LeNet-5 sử dụng các bộ lọc tích chập nhỏ để trích xuất đặc trưng cục bộ từ hình ảnh, sau đó giảm kích thước không gian qua các tầng gộp, cuối cùng sử dụng tầng kết nối đầy đủ để phân loại.



Hình 2.16 Kiến trúc của LeNet-5

Nguồn: <https://www.geeksforgeeks.org/lenet-5-architecture/>

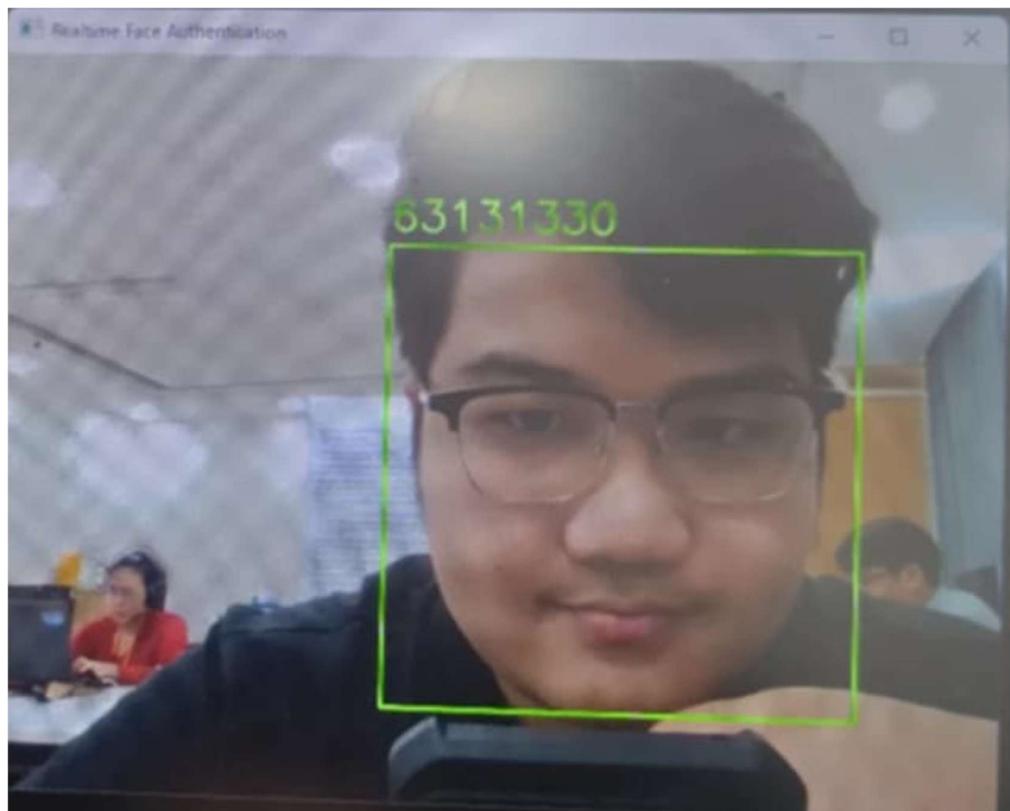
Mô hình này có số lượng tham số rất thấp so với các mô hình hiện đại, giúp nó dễ dàng huấn luyện và triển khai trên các thiết bị phần cứng hạn chế vào thời điểm đó. LeNet-5 phù hợp với các bài toán đơn giản như nhận diện chữ số hoặc ký tự, nhưng hạn chế của nó nằm ở độ sâu và khả năng học các đặc trưng phức tạp, khiến nó không hiệu quả khi áp dụng vào các bài toán hiện đại như nhận diện khuôn mặt hoặc phân loại hình ảnh phức tạp. Dù vậy, LeNet-5 vẫn là nền tảng quan trọng cho sự phát triển của các mô hình CNN sau này, cung cấp ý tưởng cơ bản về cách tổ chức các tầng tích chập và gộp trong một mạng nơ-ron.

2.4 FACENET

2.4.1 Nguồn gốc FaceNet

FaceNet là một mô hình mạng nơ-ron tích chập (CNN) tiên tiến, được phát triển bởi nhóm nghiên cứu tại Google, dẫn đầu bởi Florian Schroff, Dmitry Kalenichenko, và James Philbin. Mô hình này được giới thiệu lần đầu tiên vào năm 2015 trong bài báo “*FaceNet: A Unified Embedding for Face Recognition and Clustering*”, công bố tại Hội nghị về Thị giác Máy tính và Nhận dạng Hình ảnh (CVPR).

FaceNet ra đời với mục tiêu giải quyết các thách thức trong nhận diện khuôn mặt, đặc biệt là việc tạo ra một biểu diễn số duy nhất (embedding) cho mỗi khuôn mặt, giúp so sánh và nhận diện hiệu quả hơn. Mô hình này nhanh chóng trở thành một bước ngoặt trong lĩnh vực nhận diện khuôn mặt nhờ hiệu suất vượt trội, đạt độ chính xác cao trên các tập dữ liệu chuẩn như LFW (Labeled Faces in the Wild) với tỷ lệ chính xác lên tới 99,63%. FaceNet đã đặt nền móng cho nhiều ứng dụng thực tế, từ hệ thống xác thực danh tính, giám sát an ninh, đến các hệ thống điểm danh tự động, bao gồm cả các ứng dụng liên quan đến điểm danh sinh viên như trong đề tài này.

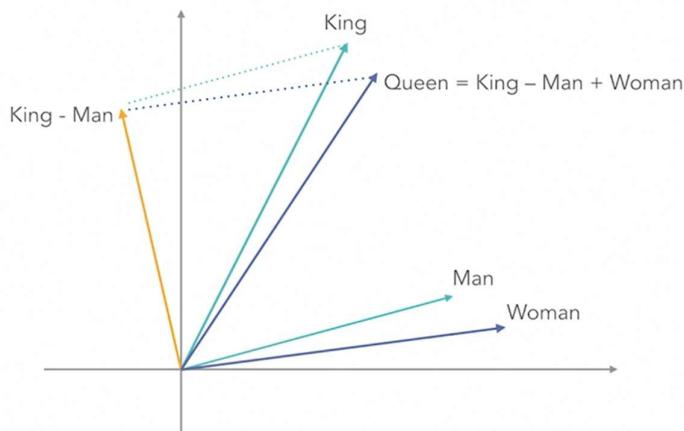


Hình 2.17 Nhận diện khuôn mặt bằng FaceNet

2.4.2 Các khái niệm cơ bản

2.4.2.1 Vector Embedding

Vector Embedding là một vector có số chiều cố định, thường nhỏ hơn so với các vector đặc trưng thông thường, được học trong quá trình huấn luyện để biểu diễn tập hợp các đặc trưng quan trọng phục vụ cho việc phân loại các đối tượng trong không gian đã được biến đổi. Vector này đóng vai trò quan trọng trong việc tìm kiếm các điểm gần nhất (Nearest Neighbor) trong một cụm dữ liệu, dựa trên khoảng cách hoặc mối quan hệ giữa các vector embedding. Điều này cho phép so sánh hiệu quả giữa các đối tượng, đặc biệt hữu ích trong các bài toán nhận diện khuôn mặt, khi cần xác định sự tương đồng giữa các khuôn mặt dựa trên biểu diễn số học.



Hình 2.18 Vector Embedding

Nguồn: <https://medium.com/@dilip.voleti/classification-using-word2vec-b1d79d375381>

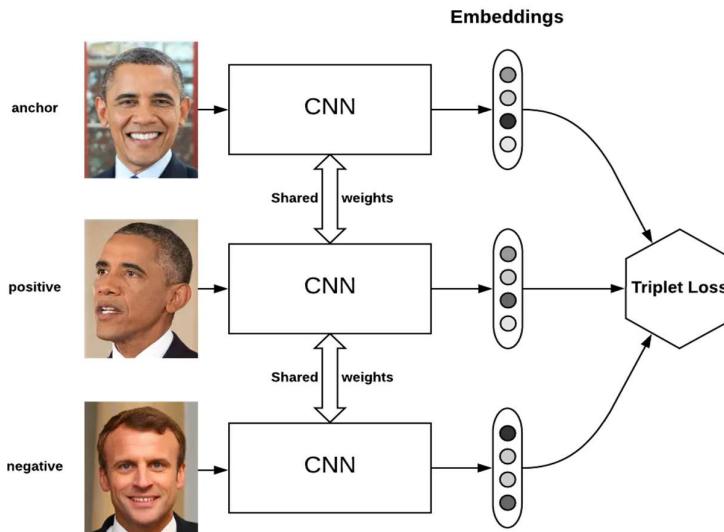
2.4.2.2 Inception V1

Inception V1, còn được biết đến với tên gọi GoogleNet, là một kiến trúc mạng CNN được Google giới thiệu vào năm 2014, nổi bật với các khối Inception đặc trưng. Các khối này cho phép mạng học theo cơ chế song song, nghĩa là một đầu vào có thể được xử lý đồng thời qua nhiều tầng tích chập khác nhau với các kích thước bộ lọc đa dạng, sau đó các kết quả được ghép nối (concatenate) thành một đầu ra duy nhất.

Phương pháp học song song này giúp mạng trích xuất được nhiều đặc trưng hơn, từ các chi tiết cục bộ đến các đặc trưng tổng quát, so với các mạng CNN truyền thống chỉ sử dụng một luồng xử lý duy nhất. Ngoài ra, Inception V1 còn áp dụng các tầng tích chập 1x1 để giảm số chiều của dữ liệu, từ đó giảm đáng kể khối lượng tính toán và tăng tốc độ huấn luyện, đồng thời giảm nguy cơ quá khớp (overfitting) khi xử lý các tập dữ liệu lớn.

2.4.2.3 Hàm mất mát Triplet Loss

Trong các bài toán nhận dạng khuôn mặt, đặc biệt là khi sử dụng phương pháp học biểu diễn (representation learning), mục tiêu là ánh xạ mỗi ảnh khuôn mặt thành một vector trong không gian đặc trưng sao cho những ảnh cùng một người thì gần nhau, và những ảnh khác người thì cách xa nhau. Một trong những hàm mất mát phổ biến được sử dụng để đạt được mục tiêu này là Triplet Loss.



Hình 2.19 Cách hoạt động của hàm Triplet Loss

Nguồn: <https://medium.com/visionwizard/demystifying-an-important-loss-function-everyone-should-know-about-triplet-loss-7c93e287e93c>

Hàm mất mát Triplet Loss được giới thiệu trong mô hình FaceNet (Google, 2015), hoạt động dựa trên ba đầu vào:

- + Anchor (A): Ảnh gốc cần xác định danh tính.
- + Positive (P): Ảnh khác của cùng một người với Anchor.
- + Negative (N): Ảnh của người khác với Anchor.

Mục tiêu của Triplet Loss là đảm bảo rằng khoảng cách giữa Anchor và Positive nhỏ hơn khoảng cách giữa Anchor và Negative ít nhất một khoảng cách biên α (margin).

Điều này được thể hiện bằng bất đẳng thức sau:

$$|f(A) - f(P)|_2^2 + \alpha < |f(A) - f(N)|_2^2$$

Trong đó:

- + $f(x)$ là hàm ánh xạ từ ảnh đầu vào sang vector đặc trưng (do mạng học được)
- + $|\cdot|_2$ là chuẩn Euclid (L2 norm).

Hàm mất mát Triplet Loss được định nghĩa như sau:

$$\mathcal{L} = \max(|f(A) - f(P)|_2^2 - |f(A) - f(N)|_2^2 + \alpha, 0)$$

Nếu khoảng cách giữa Anchor và Positive đã đủ nhỏ, và giữa Anchor và Negative đã đủ lớn, thì hàm mất mát sẽ bằng 0. Ngược lại, nếu điều kiện không được thỏa, mô hình sẽ bị phạt bằng một giá trị dương.

Triplet Loss giúp mạng học được một không gian đặc trưng mà trong đó:

- + Các vector đại diện cho cùng một người sẽ gần nhau.
- + Các vector của những người khác nhau sẽ tách biệt nhau rõ rệt.

Điều này rất phù hợp với các bài toán xác thực khuôn mặt (face verification) hoặc nhận dạng khuôn mặt (face identification) mà không cần huấn luyện mô hình phân loại theo từng nhãn cố định.

2.5 GOOGLE ML KIT FACE DETECTION

Google ML Kit là một bộ công cụ học máy do Google phát triển, cung cấp các API tiện lợi để tích hợp các tính năng học máy vào ứng dụng di động, đặc biệt trên nền tảng Android. Trong đó, API phát hiện khuôn mặt (Face Detection API) là một công cụ mạnh mẽ, cho phép ứng dụng nhận diện khuôn mặt trong ảnh hoặc video theo thời gian thực mà không phụ thuộc vào kết nối mạng. Điểm nổi bật của API này là khả năng xử lý trực tiếp trên thiết bị, mang lại tốc độ cao, bảo mật dữ liệu và sự thuận tiện khi triển khai mà không cần huấn luyện lại mô hình.

Các tính năng chính của API phát hiện khuôn mặt trong Google ML Kit bao gồm:

- + Phát hiện vị trí khuôn mặt trong khung hình và trả về tọa độ hộp giới hạn (bounding box).
- + Xác định các điểm mốc quan trọng như mắt, mũi, miệng trên khuôn mặt.
- + Nhận diện biểu cảm đơn giản, chẳng hạn như cười hoặc nhảm mặt.
- + Theo dõi chuyển động khuôn mặt liên tục qua các khung hình, phù hợp với xử lý video trực tiếp.

Những khả năng này cho phép ứng dụng phát hiện khuôn mặt một cách nhanh chóng và chính xác, kể cả trong môi trường ánh sáng không đồng đều hay khi có nhiều người xuất hiện cùng lúc. Tuy nhiên, API này chỉ tập trung vào việc phát hiện mà không có chức năng nhận diện danh tính, đòi hỏi sự kết hợp với các mô hình khác để đáp ứng nhu cầu cao hơn.



Hình 2.20 Nhận diện và xác định các đặc điểm trên khuôn mặt bằng Google ML Kit

Nguồn: <https://vncoder.vn/bai-viet/firebase-ml-kit-nhan-dien-khuon-mat-face-detection-trong-android>

API của Google ML Kit Face Detection mang lại nhiều lợi ích thiết thực. Trước hết, nó được tối ưu hóa cho thiết bị di động với mô hình nhẹ, đảm bảo tốc độ xử lý nhanh. Thứ hai, dữ liệu được xử lý cục bộ trên thiết bị, không cần gửi lên máy chủ, giúp tăng cường bảo mật thông tin. Cuối cùng, việc tích hợp vào ứng dụng Android rất đơn giản nhờ hỗ trợ các ngôn ngữ lập trình phổ biến như Kotlin và Java, kèm theo tài liệu hướng dẫn chi tiết từ Google. Mặc dù vậy, hạn chế lớn nhất là nó không thể xác định danh tính của người trong ảnh, yêu cầu bổ sung các công cụ như FaceNet để hoàn thiện quy trình nhận diện.

Trong bối cảnh ứng dụng thực tế, chẳng hạn như hệ thống điểm danh sinh viên, Google ML Kit đóng vai trò quan trọng ở giai đoạn tiền xử lý. Nó trích xuất thông tin khuôn mặt từ camera hoặc hình ảnh, sau đó cung cấp dữ liệu đầu vào cho các mô hình nhận diện như FaceNet để tạo vector đặc trưng và xác định danh tính. Nhờ sự dễ dàng tích hợp và hiệu suất ổn định, API này trở thành một giải pháp lý tưởng cho các ứng dụng di động cần phát hiện khuôn mặt theo thời gian thực.

2.6 CÔNG CỤ ĐƯỢC SỬ DỤNG

Trong quá trình phát triển, các công cụ được lựa chọn đóng vai trò quan trọng trong việc hiện thực hóa ý tưởng và đảm bảo hiệu quả của hệ thống. Những công cụ này bao gồm các ngôn ngữ lập trình, thư viện hỗ trợ, và môi trường phát triển, được chọn dựa trên tính linh hoạt, khả năng xử lý dữ liệu phức tạp, và sự phù hợp với các yêu cầu kỹ thuật của bài toán.

2.6.1 Python:

Python được chọn làm ngôn ngữ lập trình có những đặc tính vượt trội, phù hợp với các yêu cầu kỹ thuật của ứng dụng. Với cú pháp đơn giản và dễ hiểu, Python không chỉ giúp người lập trình nhanh chóng triển khai ý tưởng mà còn hỗ trợ mạnh mẽ trong các lĩnh vực như trí tuệ nhân tạo, học sâu và xử lý dữ liệu. Sự linh hoạt và hệ sinh thái thư viện phong phú của Python đã tạo điều kiện thuận lợi để xây dựng một hệ thống điểm danh sinh viên hiệu quả, từ khâu xử lý ảnh đến huấn luyện mô hình.

Những lý do chính khiến Python trở thành lựa chọn hàng đầu:

- + **Dễ học và ứng dụng linh hoạt:** Cú pháp rõ ràng của Python cho phép người dùng, kể cả những người mới bắt đầu, dễ dàng viết và kiểm tra mã. Điều này đặc biệt hữu ích khi cần thử nghiệm nhanh các thuật toán nhận diện khuôn mặt trong thời gian thực.
- + **Hỗ trợ học sâu hiệu quả:** Các thư viện như TensorFlow, Keras và PyTorch cung cấp nền tảng vững chắc để thiết kế và huấn luyện các mô hình như FaceNet. Chúng giúp tối ưu hóa hiệu suất nhận diện, đồng thời giảm độ phức tạp trong quá trình phát triển.
- + **Xử lý dữ liệu mạnh mẽ:** NumPy và Pandas là hai thư viện quan trọng hỗ trợ tổ chức và xử lý dữ liệu hình ảnh – bước chuẩn bị không thể thiếu trước khi đưa vào mô hình học sâu.
- + **Tích hợp với OpenCV:** Python kết hợp tốt với OpenCV, một công cụ chuyên dụng cho xử lý hình ảnh và video. Nhờ đó, các tác vụ như phát hiện khuôn mặt hay phân tích dữ liệu video được thực hiện mượt mà, đáp ứng yêu cầu điểm danh chính xác.

Ngoài ra, Python còn mang lại lợi thế nhờ khả năng tương thích với các công cụ phát triển hiện đại. Chẳng hạn, Jupyter Notebook cho phép lập trình viên viết mã, kiểm tra kết quả và trực quan hóa dữ liệu ngay lập tức. Điều này rất tiện lợi khi cần thử nghiệm mô hình với nhiều tập dữ liệu khác nhau hoặc tinh chỉnh tham số trong môi trường thực tế. Hơn nữa, Python hỗ trợ tích hợp với các dịch vụ đám mây như Google Colab, tận dụng sức mạnh tính toán từ xa để xử lý các tác vụ nặng mà không cần đầu tư phần cứng cục bộ đắt đỏ.

Tuy nhiên, Python cũng có hạn chế về tốc độ thực thi, thường chậm hơn so với các ngôn ngữ như C++ trong những trường hợp đòi hỏi hiệu suất cao. Dù vậy, nhược

điểm này có thể được khắc phục bằng cách tối ưu hóa mã nguồn hoặc sử dụng các thư viện được biên dịch sẵn. Sự phổ biến của Python trong cộng đồng lập trình viên, cùng với tài liệu phong phú và cập nhật liên tục, giúp việc xử lý vấn đề kỹ thuật trở nên nhanh chóng. Python không chỉ là công cụ lập trình mà còn đóng vai trò cầu nối giữa các công nghệ, từ xử lý ảnh đến triển khai học sâu, góp phần xây dựng hệ thống nhận diện khuôn mặt hiệu quả và đáng tin cậy.

2.6.2 OpenCV trong Python

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở được thiết kế để hỗ trợ các tác vụ trong lĩnh vực thị giác máy tính, tập trung vào xử lý hình ảnh và video. Đây là một công cụ mạnh mẽ cung cấp hàng loạt thuật toán và hàm tối ưu hóa, giúp lập trình viên thực hiện các nhiệm vụ phức tạp như phát hiện đối tượng, nhận diện đặc trưng và phân tích dữ liệu hình ảnh. Khi tích hợp với Python, OpenCV tận dụng được sự đơn giản và linh hoạt của ngôn ngữ này, tạo điều kiện thuận lợi cho việc phát triển các ứng dụng thị giác máy tính hiệu quả, dễ đọc và dễ bảo trì.

2.6.2.1 Sơ lược về OpenCV

OpenCV được khởi xướng bởi Intel vào năm 1999, với mục tiêu ban đầu là cung cấp một nền tảng chung cho các nhà nghiên cứu và lập trình viên trong lĩnh vực thị giác máy tính. Trải qua nhiều năm phát triển, thư viện này đã mở rộng đáng kể và hiện hỗ trợ nhiều ngôn ngữ lập trình, trong đó Python nổi bật nhờ tính dễ sử dụng và cộng đồng phát triển lớn mạnh. OpenCV chứa đựng hàng trăm hàm và thuật toán được tối ưu hóa, cho phép thực hiện các thao tác như chuyển đổi màu sắc, làm mịn ảnh, phát hiện cạnh và nhận diện đối tượng. Nhờ những đặc điểm này, OpenCV trở thành một công cụ quan trọng trong việc xử lý và phân tích hình ảnh.

Các tính năng chính của OpenCV bao gồm:

- + Phát hiện khuôn mặt: OpenCV cung cấp các phương pháp như Haar Cascade và DNN (Deep Neural Networks). Haar Cascade là một kỹ thuật truyền thống dựa trên việc phân tích các đặc trưng Haar để phát hiện vùng khuôn mặt, trong khi DNN tận dụng các mô hình học sâu để đạt độ chính xác cao hơn trong các điều kiện phức tạp.
- + Xử lý ảnh thời gian thực: Thư viện hỗ trợ xử lý video liên tục thông qua các hàm như cv2.VideoCapture để truy cập dữ liệu từ camera và cv2.imshow để hiển thị kết quả ngay lập tức, phù hợp với các ứng dụng yêu cầu phản hồi nhanh.

- + Tiền xử lý ảnh: OpenCV tích hợp các kỹ thuật như điều chỉnh độ sáng/tương phản, làm mịn ảnh bằng bộ lọc Gaussian Blur, hoặc chuyển đổi sang không gian màu Grayscale, nhằm nâng cao chất lượng ảnh trước khi áp dụng các bước phân tích tiếp theo.
- + Trích xuất đặc trưng: Các thuật toán như HOG (Histogram of Oriented Gradients) và SIFT (Scale-Invariant Feature Transform) được hỗ trợ để trích xuất các đặc điểm quan trọng từ hình ảnh, phục vụ cho việc nhận diện và phân loại đối tượng. OpenCV nổi bật nhờ khả năng tích hợp mượt mà với Python và các thư viện học sâu như TensorFlow hoặc PyTorch, tạo ra một quy trình liền mạch từ xử lý ảnh cơ bản đến phân tích nâng cao. Các hàm của OpenCV được viết bằng C++ và biên dịch sẵn, đảm bảo hiệu suất cao trên nhiều nền tảng phần cứng.

Tuy nhiên, một số phương pháp truyền thống như Haar Cascade có thể kém hiệu quả trong điều kiện ánh sáng yếu hoặc khi đối tượng bị che khuất. Để khắc phục, các kỹ thuật hiện đại hơn như MTCNN (Multi-task Cascaded Convolutional Networks) thường được kết hợp để cải thiện độ chính xác. Dù vậy, với tính linh hoạt và khả năng tùy chỉnh, OpenCV vẫn là một công cụ nền tảng không thể thiếu trong lĩnh vực thị giác máy tính.

2.6.2.2 Ứng dụng của OpenCV trong Python

OpenCV trong Python được sử dụng rộng rãi trong nhiều lĩnh vực nhờ khả năng xử lý hình ảnh và video mạnh mẽ, dễ dàng tích hợp và triển khai. Sự kết hợp giữa OpenCV và Python tạo ra một công cụ linh hoạt, giúp các nhà phát triển thực hiện các tác vụ thị giác máy tính phức tạp một cách hiệu quả. Sau đây là các ứng dụng chính của OpenCV trong Python:

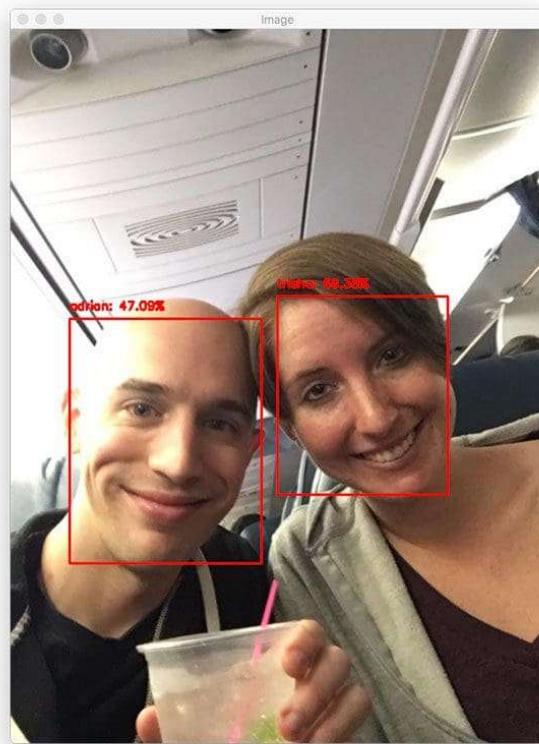
- + Xử lý ảnh cơ bản: OpenCV cung cấp các hàm để thực hiện các thao tác như thay đổi kích thước, cắt, xoay và lọc ảnh. Những kỹ thuật này thường được dùng để tiền xử lý dữ liệu hình ảnh trước khi đưa vào các mô hình học máy hoặc học sâu.
- + Phát hiện đối tượng: OpenCV hỗ trợ phát hiện đối tượng thông qua các kỹ thuật như Haar Cascade và mô hình học sâu, cho phép nhận diện các đối tượng như khuôn mặt, người đi bộ, phương tiện giao thông trong ảnh và video. Ứng dụng này rất hữu ích trong các hệ thống giám sát và an ninh.
- + Trích xuất đặc trưng: Các thuật toán như SIFT, SURF và ORB trong OpenCV giúp trích xuất đặc trưng từ ảnh, phục vụ cho các tác vụ như so sánh ảnh, nhận diện đối tượng và tái tạo 3D.

- + Phân đoạn ảnh: OpenCV cung cấp các thuật toán để chia ảnh thành các vùng khác nhau dựa trên màu sắc, kết cấu hoặc đặc trưng khác, hỗ trợ tách nền và tiền cảnh hoặc xác định đối tượng cụ thể trong ảnh.
- + Phân tích video: OpenCV cho phép phát hiện chuyển động, theo dõi đối tượng di chuyển và phân tích luồng video thời gian thực, phù hợp với các hệ thống giám sát và ứng dụng tự động hóa.
- + Tích hợp với học máy: OpenCV tương thích tốt với các thư viện như scikit-learn và TensorFlow, giúp xây dựng các quy trình thị giác máy tính toàn diện, từ xử lý ảnh đến huấn luyện mô hình.
- + Phát triển giao diện người dùng (GUI): OpenCV cung cấp các hàm để tạo giao diện đồ họa hiển thị ảnh, video và các yếu tố tương tác, hỗ trợ xây dựng ứng dụng thân thiện với người dùng.
- + Ứng dụng thời gian thực: Với hiệu suất cao, OpenCV phù hợp cho các ứng dụng yêu cầu phản hồi nhanh như hệ thống giám sát, phương tiện tự hành và thực tế ảo tăng cường.
- + Giáo dục và nghiên cứu: OpenCV được sử dụng phổ biến trong học thuật để giảng dạy các khái niệm thị giác máy tính và thực hiện nghiên cứu chuyên sâu.

Những ứng dụng trên cho thấy OpenCV trong Python là một công cụ đa năng, đáp ứng được nhiều nhu cầu từ cơ bản đến nâng cao trong lĩnh vực thị giác máy tính.

2.6.3 Google Colab

Google Colab là một nền tảng dựa trên đám mây cho phép người dùng viết và thực thi mã Python trong môi trường Jupyter Notebook. Nền tảng này đặc biệt hữu ích cho các dự án học máy và khoa học dữ liệu nhờ cung cấp quyền truy cập miễn phí vào các tài nguyên tính toán mạnh mẽ như GPU và TPU. Trong đồ án này, Google Colab được sử dụng như một công cụ quan trọng để phát triển và huấn luyện mô hình nhận diện khuôn mặt, hỗ trợ các tác vụ như tiền xử lý dữ liệu, huấn luyện mô hình, đánh giá hiệu suất và điều chỉnh siêu tham số.



Hình 2.21 Nhận diện khuôn mặt bằng OpenCV

Các tính năng chính của Google Colab:

- + Truy cập miễn phí GPU và TPU: Hỗ trợ huấn luyện các mô hình học sâu như FaceNet, vốn đòi hỏi sức mạnh tính toán lớn.
- + Hợp tác thời gian thực: Cho phép nhiều thành viên trong nhóm cùng làm việc trên một notebook, rất phù hợp cho dự án nhóm.
- + Tích hợp với Google Drive: Dễ dàng lưu trữ và truy cập notebook cùng dữ liệu dự án.
- + Thư viện cài đặt sẵn: Bao gồm các thư viện phổ biến như TensorFlow, PyTorch và OpenCV, giúp tiết kiệm thời gian cài đặt.
- + Chia sẻ dễ dàng: Notebook có thể được chia sẻ qua liên kết, cho phép người khác chạy mã mà không cần cài đặt thêm phần mềm.

Google Colab là một công cụ đa năng, hỗ trợ nhiều lĩnh vực khác nhau nhờ khả năng cung cấp tài nguyên tính toán mạnh mẽ và giao diện thân thiện. Dưới đây là các vai trò nổi bật của Colab:

- + Học máy và trí tuệ nhân tạo: Colab cho phép người dùng khai thác sức mạnh của GPU và TPU để huấn luyện các mô hình học máy và AI phức tạp mà không cần

đầu tư vào phần cứng đắt đỏ, trở thành lựa chọn lý tưởng cho các chuyên gia và nhà phát triển.

- + Phân tích dữ liệu: Nền tảng này hỗ trợ các nhà khoa học dữ liệu xử lý dữ liệu nhanh chóng, thực hiện phân tích chuyên sâu và tạo ra các biểu đồ trực quan, giúp tối ưu hóa quy trình làm việc.
- + Nghiên cứu khoa học: Các nhà nghiên cứu có thể tận dụng Colab để lập trình mô phỏng, phân tích dữ liệu số và chia sẻ kết quả nghiên cứu một cách dễ dàng, nhờ môi trường làm việc tích hợp và khả năng cộng tác.
- + Giáo dục: Colab đóng vai trò quan trọng trong việc giảng dạy và học tập, cung cấp cho giáo viên và học sinh một công cụ hiệu quả để khám phá các khái niệm về lập trình, học máy và phân tích dữ liệu thông qua thực hành trực tiếp.

Ưu điểm và hạn chế của Google Collab:

Ưu điểm:

- + Cung cấp tài nguyên tính toán miễn phí và mạnh mẽ.
- + Hỗ trợ làm việc nhóm hiệu quả thông qua tính năng hợp tác.
- + Tích hợp tốt với Google Drive và các thư viện học máy.

Hạn chế:

- + Tài nguyên bị giới hạn về thời gian sử dụng, phiên làm việc có thể bị ngắt sau thời gian không hoạt động.
- + Phụ thuộc vào kết nối internet ổn định.
- + Có thể tiềm ẩn rủi ro bảo mật khi xử lý dữ liệu nhạy cảm.

2.6.4 Jetpack Compose

Jetpack Compose là một bộ công cụ hiện đại do Google phát triển dành cho việc xây dựng giao diện người dùng (UI) trên nền tảng Android, nhằm mục đích tối ưu hóa và đơn giản hóa quy trình thiết kế ứng dụng. Được giới thiệu lần đầu dưới dạng bản thử nghiệm tại sự kiện Google I/O 2019 và chính thức ổn định vào tháng 7 năm 2021, Jetpack Compose đã nhanh chóng trở thành một giải pháp phổ biến cho các nhà phát triển Android. Sử dụng ngôn ngữ lập trình Kotlin, Jetpack Compose áp dụng mô hình lập trình khai báo (declarative programming) và dựa trên cách tiếp cận phản ứng (reactive), cho phép các lập trình viên mô tả giao diện trực tiếp bằng mã Kotlin thông qua các hàm composable thay vì sử dụng XML như cách truyền thống.

Jetpack Compose mang lại nhiều ưu điểm nổi bật, giúp cải thiện hiệu quả phát triển giao diện người dùng:

- **Lập trình khai báo:** Với Jetpack Compose, giao diện được xây dựng thông qua các hàm composable như Text, Button, hay Row. Khi trạng thái của ứng dụng thay đổi, Compose tự động cập nhật giao diện mà không cần lập trình viên phải viết mã điều khiển thủ công. Điều này giúp giảm sự phức tạp và tăng tính trực quan trong việc quản lý giao diện.
- **Tích hợp với Kotlin:** Được xây dựng hoàn toàn bằng Kotlin, Jetpack Compose loại bỏ sự bất tiện khi phải chuyển đổi giữa XML và mã lập trình, giúp mã nguồn trở nên thống nhất và dễ bảo trì. Các tính năng của Kotlin như lambda và coroutines cũng được tận dụng để tăng tính linh hoạt khi phát triển.
- **Hỗ trợ Material Design:** Jetpack Compose tích hợp sẵn các thành phần Material Design (bao gồm Material Design 3), giúp nhà phát triển dễ dàng tạo ra giao diện đẹp, đồng bộ và tuân thủ tiêu chuẩn thiết kế của Google. Ví dụ, các thành phần như nút, thanh điều hướng, hoặc danh sách có thể được thiết kế nhanh chóng với phong cách Material.
- **Hiệu suất tối ưu:** Jetpack Compose sử dụng cơ chế tái cấu trúc (recomposition) thông minh, chỉ cập nhật những thành phần giao diện bị ảnh hưởng bởi sự thay đổi trạng thái, thay vì làm mới toàn bộ giao diện. Điều này giúp cải thiện hiệu suất, đặc biệt trên các ứng dụng có giao diện động và phức tạp.
- **Tương thích linh hoạt:** Jetpack Compose có thể hoạt động song song với hệ thống View truyền thống, cho phép tích hợp dần vào các dự án hiện có mà không cần viết lại toàn bộ mã. Điều này giúp các nhóm phát triển chuyển đổi sang Compose một cách dễ dàng và ít rủi ro.

Dưới đây là một ví dụ minh họa cách sử dụng Jetpack Compose để tạo một giao diện đơn giản hiển thị một tiêu đề và một nút bấm:

```

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            TestttTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) {
                    innerPadding ->
                        val context = this
                        Greeting(
                            name = "Vu Minh Thinh",
                            modifier =
                        Modifier.padding(innerPadding),
                            context = context
                        )
                }
            }
        }
    }

    @Composable
    fun Greeting(name: String, modifier: Modifier = Modifier,
    context: android.content.Context) {
        Column(modifier = modifier.padding(16.dp)) {
            Text(text = "Hello $name!")
            Spacer(modifier = Modifier.height(8.dp))
            ToastButton(context = context)
        }
    }

    @Composable
    fun ToastButton(context: android.content.Context) {
        Button(
            onClick = {
                Toast.makeText(context, "Button Clicked!!",
                Toast.LENGTH_SHORT).show()
            },
            modifier = Modifier.padding(16.dp)
        ) {
            Text(text = "Nhấn vào đây")
        }
    }

    @Preview(showBackground = true)
    @Composable
    fun GreetingPreview() {
        TestttTheme {
            Greeting(name = "Android", context =
        LocalContext.current)
        }
    }
}

```

Kết quả của đoạn code trên được hiển thị như hình dưới:



Hình 2.22 Một ví dụ về Jetpack Compose

2.7 LFW DATASET (LABELED FACES IN THE WILD)

Bộ dữ liệu LFW (Labeled Faces in the Wild) là một trong những tài nguyên quan trọng được sử dụng rộng rãi trong lĩnh vực nhận diện khuôn mặt, đóng vai trò như một tiêu chuẩn để đánh giá hiệu suất của các mô hình học máy và học sâu. Được phát triển bởi Đại học Massachusetts Amherst, bộ dữ liệu này cung cấp một tập hợp lớn các hình ảnh khuôn mặt từ nhiều nguồn khác nhau, phản ánh sự đa dạng trong điều kiện chụp ảnh thực tế. Với mục tiêu hỗ trợ nghiên cứu và phát triển các thuật toán nhận diện khuôn mặt, LFW trở thành công cụ không thể thiếu trong việc kiểm tra độ chính xác và khả năng thích nghi của các mô hình, từ đó góp phần thúc đẩy tiến bộ trong công nghệ thị giác máy tính.



Hình 2.23 Bộ dữ liệu Labeled Faces in the Wild (LFW)

Nguồn: <https://www.kaggle.com/datasets/atulanandjha/lfwpeople>

2.7.1 Nguồn gốc và mục đích

LFW được giới thiệu lần đầu tiên vào năm 2007 bởi các nhà nghiên cứu tại Đại học Massachusetts Amherst, với ý tưởng tạo ra một bộ dữ liệu thực tế để giải quyết các thách thức trong nhận diện khuôn mặt ngoài môi trường kiểm soát. Khác với các tập dữ liệu được chụp trong điều kiện lý tưởng, LFW tập trung vào các ảnh được lấy từ internet, chủ yếu từ các trang tin tức, mang lại sự phong phú về góc quay, ánh sáng, biểu cảm và chất lượng hình ảnh. Mục đích chính của bộ dữ liệu này là cung cấp một nền tảng để thử nghiệm và cải thiện các thuật toán nhận diện khuôn mặt trong các tình huống phức tạp, đồng thời đánh giá khả năng phân biệt giữa các cá nhân trong điều kiện tự nhiên. Điều này giúp các nhà nghiên cứu hiểu rõ hơn về giới hạn của công nghệ và định hướng phát triển các giải pháp thực tiễn.

2.7.2 Cấu trúc và nội dung

Bộ dữ liệu LFW bao gồm hơn 13.000 ảnh của 5.749 cá nhân, với mỗi người có từ 1 đến 530 hình ảnh, tùy thuộc vào mức độ nổi tiếng của họ. Các ảnh này được chụp trong nhiều ngữ cảnh khác nhau, từ ảnh chân dung đến ảnh chụp nhóm, với sự biến thiên lớn về góc độ, ánh sáng, và chất lượng. Kích thước hình ảnh dao động từ 250x250 pixel trở lên, nhưng thường được chuẩn hóa về 250x250 pixel để đảm bảo tính đồng nhất khi sử dụng. Một đặc điểm nổi bật là LFW cung cấp các cặp ảnh đã được ghi nhãn (labeled pairs), trong đó một số cặp là của cùng một người, một số khác là của các cá nhân khác nhau. Điều này cho phép đánh giá hiệu quả của các mô hình thông qua các nhiệm vụ như xác minh khuôn mặt (verification) và nhận diện danh tính (identification). Ngoài ra, LFW cũng bao gồm các tập con như “View 1” và “View 2” để kiểm tra và huấn luyện, giúp đảm bảo tính minh bạch trong quá trình đánh giá.

2.7.3 Ưu điểm và hạn chế

Ưu điểm của LFW:

- Đa dạng thực tế: Với nguồn gốc từ internet, LFW phản ánh chính xác các thách thức trong môi trường tự nhiên, bao gồm sự thay đổi về ánh sáng, góc quay và biểu cảm.
- Quy mô lớn: Số lượng ảnh và cá nhân phong phú cho phép huấn luyện các mô hình nhận diện khuôn mặt với độ tin cậy cao.

- Dữ liệu ghi nhãn rõ ràng: Các cặp ảnh được ghi nhãn chính xác giúp dễ dàng đánh giá hiệu suất mô hình thông qua các chỉ số như độ chính xác (accuracy) và tỷ lệ sai lầm (false positive rate).
- Tiêu chuẩn ngành: LFW đã trở thành một chuẩn mực chung, giúp so sánh hiệu quả giữa các thuật toán và mô hình khác nhau.

Hạn chế của LFW:

- Chất lượng ảnh không đồng đều: Do được thu thập từ nhiều nguồn, một số ảnh có độ phân giải thấp hoặc bị nhiễu, làm giảm độ chính xác trong một số trường hợp.
- Thiếu đa dạng dân tộc: Tập dữ liệu chủ yếu tập trung vào các cá nhân nổi tiếng phương Tây, dẫn đến thiếu sót trong việc đại diện cho các nhóm dân tộc khác.
- Không có ghi nhãn biểu cảm chi tiết: LFW chỉ cung cấp thông tin về danh tính, không bao gồm các đặc điểm như biểu cảm hay độ tuổi, hạn chế khả năng nghiên cứu sâu hơn.
- Cần tiền xử lý: Các ảnh cần được căn chỉnh và chuẩn hóa trước khi sử dụng, đòi hỏi thêm công sức trong quá trình chuẩn bị dữ liệu.

2.7.4 Ứng dụng trong nghiên cứu

LFW đóng vai trò quan trọng trong việc phát triển và kiểm tra các thuật toán nhận diện khuôn mặt. Bộ dữ liệu này thường được sử dụng để huấn luyện và đánh giá các mô hình như FaceNet, DeepFace và OpenFace, nhờ khả năng cung cấp các cặp ảnh đa dạng để kiểm tra khả năng phân biệt. Các nhà nghiên cứu sử dụng LFW để đo lường hiệu suất của các phương pháp xác minh khuôn mặt, trong đó mô hình phải xác định liệu hai ảnh có thuộc về cùng một người hay không. Ngoài ra, LFW còn hỗ trợ trong việc thử nghiệm các kỹ thuật tiền xử lý ảnh, như căn chỉnh khuôn mặt hoặc loại bỏ nhiễu, để cải thiện kết quả nhận diện. Nhờ đó, LFW không chỉ là một công cụ đánh giá mà còn là nguồn cảm hứng để phát triển các giải pháp nhận diện khuôn mặt tiên tiến hơn.

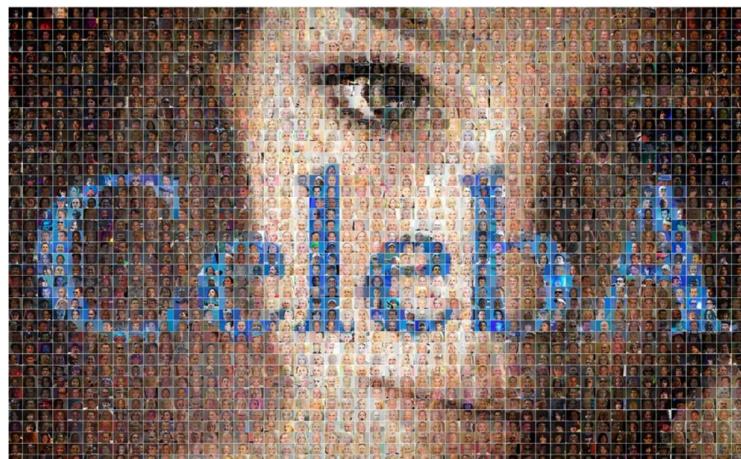
2.7.5 Tầm quan trọng và triển vọng

LFW đại diện cho một bước tiến trong việc xây dựng các bộ dữ liệu thực tế, giúp các nhà nghiên cứu vượt qua giới hạn của các tập dữ liệu nhân tạo. Sự phổ biến của LFW đã khuyến khích cộng đồng khoa học phát triển các phương pháp mới, từ các thuật toán truyền thống đến các mô hình học sâu hiện đại. Tuy nhiên, với sự phát triển của công nghệ, LFW cũng bộc lộ những hạn chế, dẫn đến nhu cầu tạo ra các bộ dữ liệu mới

như IJB-A hoặc MS-Celeb-1M, với sự đa dạng lớn hơn về dân tộc và điều kiện chụp ảnh. Trong tương lai, LFW có thể được cập nhật hoặc kết hợp với các tập dữ liệu khác để đáp ứng tốt hơn các yêu cầu nghiên cứu, đặc biệt trong việc xây dựng các hệ thống nhận diện khuôn mặt chính xác và công bằng hơn.

2.8 CELEBA DATASET (CELEBFACES ATTRIBUTES DATASET)

Bộ dữ liệu CelebA (CelebFaces Attributes Dataset) là một tài nguyên quan trọng trong lĩnh vực thị giác máy tính, đặc biệt trong các nghiên cứu liên quan đến nhận diện khuôn mặt, phân tích thuộc tính khuôn mặt và sinh ảnh (image generation). Được phát triển bởi nhóm nghiên cứu tại Đại học Trung Văn Hồng Kông (CUHK), CelebA cung cấp một tập hợp lớn các hình ảnh khuôn mặt kèm theo nhiều ghi nhãn thuộc tính, giúp hỗ trợ các nhà nghiên cứu trong việc phát triển và kiểm tra các thuật toán học máy, học sâu. Với sự đa dạng về dữ liệu và thông tin chi tiết, CelebA đã trở thành một trong những bộ dữ liệu phổ biến, đóng góp đáng kể vào sự tiến bộ của công nghệ nhận diện và phân tích khuôn mặt.



Hình 2.24 Bộ dữ liệu CelebFaces Attributes (CelebA)

Nguồn: <https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

2.8.1 Nguồn gốc và mục đích

CelebA được giới thiệu lần đầu tiên vào năm 2015 bởi nhóm nghiên cứu tại Đại học Trung Văn Hồng Kông, nhằm đáp ứng nhu cầu ngày càng tăng về dữ liệu khuôn mặt chất lượng cao cho các ứng dụng học sâu. Bộ dữ liệu này được xây dựng bằng cách thu thập hình ảnh từ internet, tập trung vào các khuôn mặt của những người nổi tiếng trên toàn thế giới, từ đó tạo nên một tập hợp đa dạng về ngoại hình và phong cách. Mục tiêu chính của CelebA là cung cấp một nền tảng để nghiên cứu các bài toán liên quan đến nhận diện khuôn mặt, phân tích thuộc tính (như giới tính, tuổi tác, kiểu tóc) và sinh

ảnh khuôn mặt (face generation). Khác với các bộ dữ liệu chỉ tập trung vào danh tính, CelebA cung cấp thêm các ghi nhãn thuộc tính phong phú, giúp mở rộng phạm vi ứng dụng của nó trong nhiều lĩnh vực nghiên cứu.

2.8.2 Cấu trúc và nội dung

Bộ dữ liệu CelebA bao gồm hơn 202.599 hình ảnh khuôn mặt của 10.177 người nổi tiếng, với mỗi người có trung bình khoảng 20 ảnh. Các hình ảnh này được thu thập từ internet và có kích thước góc đa dạng, nhưng thường được chuẩn hóa về kích thước 178x218 pixel để thuận tiện cho việc xử lý. Một đặc điểm nổi bật của CelebA là nó cung cấp 40 thuộc tính nhị phân (binary attributes) cho mỗi hình ảnh, bao gồm các đặc điểm như “đeo kính”, “có râu”, “mỉm cười”, “tóc vàng”, “đôi mũ” và “trẻ tuổi”. Ngoài ra, CelebA cũng cung cấp thông tin về vị trí của các điểm mốc (landmarks) trên khuôn mặt, chẳng hạn như mắt, mũi, miệng, giúp hỗ trợ các tác vụ như căn chỉnh khuôn mặt (face alignment). Bộ dữ liệu được chia thành ba tập con: tập huấn luyện (training), tập kiểm tra (validation) và tập đánh giá (testing), đảm bảo tính khoa học trong quá trình phát triển và đánh giá mô hình.

2.8.3 Ưu điểm và hạn chế

*Ưu điểm của CelebA:

- + Quy mô lớn và đa dạng: Với hơn 200.000 hình ảnh và 10.000 danh tính, CelebA cung cấp một lượng dữ liệu phong phú, phù hợp cho các mô hình học sâu cần huấn luyện trên tập dữ liệu lớn.
- + Ghi nhãn thuộc tính chi tiết: 40 thuộc tính nhị phân cho phép nghiên cứu nhiều khía cạnh của khuôn mặt, từ đặc điểm vật lý đến biểu cảm, mở rộng phạm vi ứng dụng của bộ dữ liệu.
- + Hỗ trợ đa dạng tác vụ: CelebA được sử dụng cho nhiều bài toán như nhận diện khuôn mặt, phân loại thuộc tính, sinh ảnh và chuyển đổi phong cách (style transfer).
- + Cung cấp điểm mốc khuôn mặt: Dữ liệu về vị trí các điểm mốc giúp cải thiện độ chính xác trong các tác vụ như căn chỉnh khuôn mặt và tái tạo khuôn mặt 3D.

*Hạn chế của CelebA:

- + Chất lượng ảnh không đồng đều: Vì được thu thập từ internet, một số ảnh có độ phân giải thấp, bị nhiễu hoặc không rõ nét, gây khó khăn trong quá trình xử lý.

- + Thiếu sự cân bằng về thuộc tính: Một số thuộc tính, ví dụ như “đội mũ” hoặc “có râu”, xuất hiện không đồng đều giữa các cá nhân, dẫn đến thiên lệch trong huấn luyện mô hình.
- + Tập trung vào người nổi tiếng: Dữ liệu chủ yếu đến từ những người nổi tiếng, thường có ngoại hình được chỉnh sửa hoặc trang điểm, không phản ánh hoàn toàn sự đa dạng của người dùng thực tế.
- + Hạn chế về tính riêng tư: Dù là dữ liệu công khai, việc sử dụng hình ảnh của người nổi tiếng có thể đặt ra các vấn đề về quyền riêng tư và đạo đức.

2.8.4 Ứng dụng trong nghiên cứu

CelebA đã tạo nên một bước tiến quan trọng trong lĩnh vực thị giác máy tính bằng cách cung cấp một bộ dữ liệu phong phú và đa năng, đáp ứng nhu cầu của các nhà nghiên cứu trong thời kỳ bùng nổ của học sâu. Sự hiện diện của các thuộc tính chi tiết và dữ liệu điểm mốc đã mở ra nhiều hướng nghiên cứu mới, từ phân tích khuôn mặt đến sinh ảnh sáng tạo. Tuy nhiên, với sự phát triển của công nghệ và yêu cầu ngày càng cao về tính đa dạng, CelebA cũng bộc lộ một số hạn chế, đặc biệt trong việc đại diện cho các nhóm dân tộc thiểu số và các điều kiện chụp ảnh phức tạp. Trong tương lai, CelebA có thể được mở rộng hoặc kết hợp với các bộ dữ liệu khác để cải thiện tính đại diện và độ chính xác, đồng thời đáp ứng các tiêu chuẩn đạo đức và quyền riêng tư ngày càng nghiêm ngặt. Bộ dữ liệu này sẽ tiếp tục là một nguồn tài nguyên quý giá, góp phần thúc đẩy các tiến bộ trong công nghệ nhận diện và phân tích khuôn mặt.

Chương 3. XÂY DỰNG CÔNG CỤ ĐIỂM DANH BẰNG NHẬN DIỆN KHUÔN MẶT

3.1 PHÂN TÍCH BÀI TOÁN

3.1.1 Mô tả bài toán

Bài toán xây dựng công cụ điểm danh bằng nhận diện khuôn mặt nhằm mục đích tự động hóa quá trình ghi nhận sự hiện diện của sinh viên trong lớp học mà không cần sử dụng giấy tờ hoặc các phương pháp thủ công. Yêu cầu chính là hệ thống phải xác định danh tính của từng sinh viên dựa trên hình ảnh khuôn mặt chụp từ camera, sau đó ghi lại thông tin điểm danh vào một cách chính xác và nhanh chóng. Điều này đòi hỏi hệ thống phải xử lý ảnh hoặc video thời gian thực, nhận diện khuôn mặt từ nhiều góc độ và điều kiện ánh sáng khác nhau.

Các yêu cầu cụ thể của bài toán:

- + Phát hiện khuôn mặt: Hệ thống phải xác định chính xác vị trí khuôn mặt trong khung hình từ camera hoặc ảnh tĩnh.
- + Nhận diện danh tính: Dựa trên đặc trưng khuôn mặt, hệ thống cần so sánh và xác định danh tính sinh viên so với cơ sở dữ liệu đã được huấn luyện.
- + Xử lý thời gian thực: Đảm bảo phản hồi nhanh chóng trong môi trường lớp học đông đúc, với khả năng xử lý nhiều khuôn mặt cùng lúc.
- + Tính chính xác cao: Giảm thiểu sai sót trong việc nhận diện sai danh tính sinh viên.
- + Bảo mật dữ liệu: Sử dụng các biện pháp mã hóa và lưu trữ an toàn để bảo vệ thông tin khuôn mặt.
- + Khả năng mở rộng: Hỗ trợ thêm danh sách sinh viên mới hoặc cập nhật dữ liệu khi cần thiết.

3.2 XÂY DỰNG MÔ HÌNH CHO NHẬN DẠNG KHUÔN MẶT

3.2.1 Tiền xử lý ảnh

Tiền xử lý ảnh là bước đầu tiên và quan trọng trong quy trình xây dựng mô hình nhận dạng khuôn mặt, nhằm chuẩn hóa dữ liệu hình ảnh để đảm bảo chất lượng đầu vào cho mô hình học sâu. Giai đoạn này giúp loại bỏ nhiễu, cải thiện độ rõ nét và đồng nhất dữ liệu, từ đó nâng cao hiệu suất nhận diện. Tiền xử lý ảnh không chỉ ảnh hưởng đến độ chính xác của mô hình mà còn giúp giảm tải tính toán, đặc biệt khi xử lý số lượng lớn ảnh trong môi trường thực tế. Dưới đây là 5 giai đoạn chính trong quá trình tiền xử lý ảnh:

Giai đoạn 1: Phát hiện khuôn mặt

Quá trình bắt đầu bằng việc sử dụng thuật toán Haar Cascade để xác định vị trí khuôn mặt trong ảnh thông qua việc phân tích ảnh grayscale. Phương pháp này phát hiện các vùng khuôn mặt dựa trên các đặc trưng như hình dạng và cấu trúc, sau đó trích xuất tọa độ của vùng khuôn mặt đầu tiên. Tuy nhiên, bước căn chỉnh dựa trên các điểm mốc (như mắt, mũi, miệng) để điều chỉnh hướng và tỷ lệ chưa được thực hiện, do đó hiệu suất có thể bị ảnh hưởng nếu ảnh có góc quay hoặc tư thế khác nhau.

Giai đoạn 2: Chuẩn hóa kích thước ảnh

Sau khi phát hiện, vùng khuôn mặt được resize về kích thước cố định 224x224 pixel để phù hợp với yêu cầu đầu vào của mô hình FaceNet. Bước này đảm bảo tính đồng nhất trong dữ liệu, giúp mô hình xử lý nhanh hơn và cải thiện hiệu quả huấn luyện cũng như nhận diện.

Giai đoạn 3: Chuyển đổi không gian màu và chuẩn hóa giá trị pixel

Ảnh khuôn mặt được chuyển từ không gian màu BGR (mặc định của OpenCV) sang RGB để phù hợp với mô hình học sâu, thay vì chuyển sang Grayscale như một số phương pháp truyền thống. Đồng thời, giá trị pixel của ảnh được chuẩn hóa về khoảng [0, 1] bằng cách chia cho 255.0, giúp tối ưu hóa quá trình tính toán và đảm bảo dữ liệu phù hợp với đầu vào của mô hình.

Giai đoạn 4: Cắt bỏ vùng không liên quan

Sau khi phát hiện khuôn mặt, vùng khuôn mặt được cắt ra dựa trên tọa độ từ Haar Cascade, loại bỏ các khu vực không cần thiết như nền hoặc các đối tượng khác. Điều này giúp tập trung dữ liệu vào khu vực quan trọng nhất, nâng cao hiệu quả của quá trình nhận diện.

Giai đoạn 5: Tăng cường dữ liệu (Data Augmentation)

Tăng cường dữ liệu được thực hiện trong giai đoạn huấn luyện. Sử dụng ImageDataGenerator. Điều này giúp mô hình học được các đặc trưng tổng quát hơn, giảm nguy cơ quá khớp trong quá trình huấn luyện.



Hình 3.1 Ảnh ban đầu



Hình 3.2 Ảnh sau khi tiền xử lý



Hình 3.3 Ảnh ban đầu (2)



Hình 3.4 Ảnh sau khi tiền xử lý (2)

3.2.2 Xuất kết quả ra JSON

Sau khi hoàn tất tiền xử lý ảnh và huấn luyện mô hình, bước tiếp theo tập trung vào việc chuẩn bị dữ liệu hỗ trợ để tích hợp vào hệ thống nhận diện khuôn mặt. Cụ thể, quá trình này bao gồm việc tạo tệp embeddings.json từ các ảnh đã xử lý (dựa trên embeddings được trích xuất từ mô hình FaceNet) và tệp students.json từ file students.txt (chứa thông tin MSSV và tên sinh viên). Hai tệp này đóng vai trò như cơ sở dữ liệu ánh xạ, giúp hệ thống xác định danh tính của sinh viên trong quá trình nhận diện thời gian thực.

Giai đoạn 1: Tạo embeddings.json từ ảnh

Quá trình bắt đầu bằng việc sử dụng mô hình FaceNet đã huấn luyện (từ file face_auth_model_facenet.h5) để trích xuất embeddings từ các ảnh trong thư mục processed_data. File generate_embeddings.py thực hiện việc này bằng cách đọc từng ảnh, đưa qua mô hình để tạo vectơ đặc trưng (embedding), và kết hợp với MSSV tương ứng.

Kết quả được lưu tạm thời vào file embeddings.pkl. Tiếp theo, file convert_data_to_json.py tải dữ liệu từ embeddings.pkl, chuyển đổi các embedding (dạng NumPy array) thành danh sách, và tạo cấu trúc JSON với định dạng {‘array’: [{‘mssv’: mssv, ‘embedding’: [values]}]}. Tệp embeddings.json được lưu để sử dụng trong ứng dụng nhận diện, cho phép so sánh embedding của khuôn mặt mới với cơ sở dữ liệu.

Giai đoạn 2: Tạo students.json từ students.txt

Song song với việc tạo embeddings.json, file convert_data_to_json.py cũng xử lý file students.txt, nơi chứa thông tin MSSV và tên sinh viên (mỗi dòng là một cặp MSSV - tên, cách nhau bằng khoảng trắng). Dữ liệu được đọc và chuyển thành một từ điển Python, sau đó được lưu dưới dạng JSON với cấu trúc {'mssv': 'name'} trong tệp students.json. Bước này đảm bảo hệ thống có thể ánh xạ MSSV từ embeddings sang tên đầy đủ của sinh viên, tăng tính tiện lợi với người dùng khi hiển thị kết quả.

Sau khi tạo hai tệp JSON, em sử dụng Python để kiểm tra nội dung (json.load(open('embeddings.json'))) hoặc log thông tin để đảm bảo dữ liệu được chuyển đổi chính xác. Điều này giúp phát hiện sớm các lỗi như thiếu dữ liệu hoặc định dạng không hợp lệ.

3.3 XÂY DỰNG MÔ HÌNH

Phần này nói về quá trình xây dựng mô hình nhận dạng khuôn mặt, một thành phần quan trọng của hệ thống điểm danh sinh viên tự động. Quá trình bao gồm các giai đoạn từ tiền xử lý dữ liệu đầu vào, tăng cường dữ liệu, thiết kế và huấn luyện mô hình FaceNet, tạo embeddings, đến chuẩn bị dữ liệu cho nhận diện thời gian thực.

Mục tiêu là phát triển một mô hình có khả năng nhận diện danh tính sinh viên từ hình ảnh hoặc video với độ chính xác cao, đồng thời tối ưu hóa để triển khai trên các thiết bị di động như điện thoại Android. Mô hình được xây dựng dựa trên FaceNet, tích hợp các công nghệ như OpenCV, TensorFlow, nhằm đảm bảo hiệu suất và tính ứng dụng thực tiễn.

3.3.1 Tăng cường dữ liệu đã được tiền xử lí

Sau khi dữ liệu ảnh khuôn mặt được tiền xử lý (bao gồm phát hiện, cắt vùng khuôn mặt, chuẩn hóa kích thước, và chuẩn hóa giá trị pixel), dữ liệu đã sẵn sàng trong thư mục processed_data với cấu trúc phù hợp. Tuy nhiên, để mô hình học sâu có khả năng tổng quát hóa tốt và thích nghi với các điều kiện thực tế, việc tăng cường dữ liệu (Data Augmentation) là một bước cần thiết.

Giai đoạn này không được thực hiện trong bước tiền xử lý ban đầu mà được tích hợp trực tiếp vào quá trình huấn luyện, nhằm tạo ra các biến thể của dữ liệu một cách động và hiệu quả. Tăng cường dữ liệu giúp mô hình học được các đặc trưng khuôn mặt trong nhiều điều kiện khác nhau, từ đó giảm nguy cơ quá khớp (overfitting) và cải thiện hiệu suất nhận diện trong môi trường thực tế, nơi ánh sáng, góc quay, hoặc chất lượng ảnh có thể thay đổi. Trong dự án này, tăng cường dữ liệu được thực hiện thông qua thư viện ImageDataGenerator của Keras, như được triển khai trong file train_facenet.py. Các kỹ thuật được áp dụng bao gồm nhiều biến đổi hình học và quang học, với các tham số cụ thể để đảm bảo sự đa dạng của dữ liệu mà không làm mất đi tính chất nhận diện của khuôn mặt.

Dưới đây là các kỹ thuật tăng cường dữ liệu đã được sử dụng:

Xoay ảnh (rotation_range):

Ảnh được xoay ngẫu nhiên trong khoảng ± 20 độ (rotation_range=20). Điều này mô phỏng các tình huống thực tế khi sinh viên có thể nghiêng đầu trong lúc chụp ảnh, giúp mô hình học được các đặc trưng khuôn mặt từ các góc độ khác nhau.

Dịch chuyển ngang và dọc (width_shift_range và height_shift_range):

Ảnh được dịch chuyển ngẫu nhiên theo chiều ngang và dọc trong khoảng $\pm 20\%$ kích thước ảnh (width_shift_range=0.2, height_shift_range=0.2). Bước này mô phỏng trường hợp khuôn mặt không nằm chính giữa khung hình, giúp mô hình nhận diện tốt hơn khi vị trí khuôn mặt thay đổi.

Lật ngang (horizontal_flip):

Ảnh được lật ngang ngẫu nhiên (horizontal_flip=True), tạo ra phiên bản đối xứng của khuôn mặt. Điều này hữu ích vì khuôn mặt con người thường có tính đối xứng, và kỹ thuật này giúp mô hình học được các đặc trưng từ cả hai phía, tăng khả năng nhận diện trong các điều kiện đối xứng.

Thay đổi độ sáng (brightness_range):

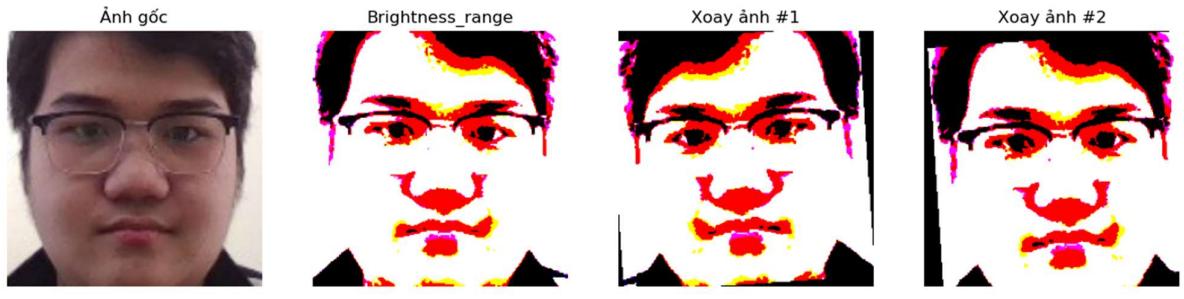
Độ sáng của ảnh được điều chỉnh ngẫu nhiên trong khoảng từ 80% đến 120% giá trị ban đầu (brightness_range=[0.8, 1.2]). Bước này mô phỏng các điều kiện ánh sáng khác nhau trong lớp học, chẳng hạn như ánh sáng yếu hoặc ánh sáng quá mạnh, đảm bảo mô hình hoạt động tốt trong môi trường thực tế.

Zoom ngẫu nhiên (zoom_range):

Ảnh được phóng to hoặc thu nhỏ ngẫu nhiên trong khoảng từ 80% đến 120% ($\text{zoom_range}=[0.8, 1.2]$). Kỹ thuật này giúp mô hình nhận diện khuôn mặt ở các khoảng cách khác nhau từ camera, chẳng hạn khi sinh viên đứng gần hoặc xa hơn so với thiết bị chụp.

Các biến đổi trên được áp dụng động thông qua ImageDataGenerator trong file `train_facenet.py`. Dữ liệu huấn luyện (80% tổng số ảnh) được đưa vào `datagen.flow` (`train_images, train_labels, batch_size=32`), tạo ra các lô dữ liệu (batch) với các biến thể ngẫu nhiên trong suốt quá trình huấn luyện. Điều này không chỉ làm tăng tính đa dạng của dữ liệu mà còn giúp mô hình học được các đặc trưng ổn định, không phụ thuộc vào các yếu tố ngoại cảnh như ánh sáng, góc chụp, hay vị trí khuôn mặt.

3.3.2 Kiến trúc mô hình



3.3.3 Huấn luyện và đánh giá mô hình

3.3.3.1 Huấn luyện mô hình

Chuẩn bị dữ liệu:

- + Dữ liệu ảnh đã được tăng cường để tăng tính đa dạng và giảm overfitting.
- + Chia dữ liệu thành 2 tập:
 - + Tập huấn luyện: 70% (dùng để huấn luyện mô hình).
 - + Tập kiểm tra (validation): 30% (dùng để điều chỉnh siêu tham số).
- + Sử dụng ImageDataGenerator để tăng cường dữ liệu trong thời gian thực.

Cấu hình huấn luyện:

- + **Hàm mất mát (loss):** Categorical Crossentropy (phù hợp cho bài toán phân loại đa lớp).
- + **Bộ tối ưu (optimizer):** Adam với learning rate 0.001.
- + **Số epoch:** 50.
- + **Batch size:** 32.

- + Sử dụng **EarlyStopping** để dừng huấn luyện nếu hiệu suất trên tập kiểm tra không cải thiện.

Quy trình huấn luyện:

- + Mô hình được huấn luyện trên tập huấn luyện, sử dụng dữ liệu tăng cường từ ImageDataGenerator.
- + Theo dõi độ chính xác (accuracy) và giá trị mất mát (loss) trên tập kiểm tra sau mỗi epoch.
- + Lưu mô hình tốt nhất dựa trên độ chính xác trên tập kiểm tra.

3.3.3.2 Đánh giá mô hình

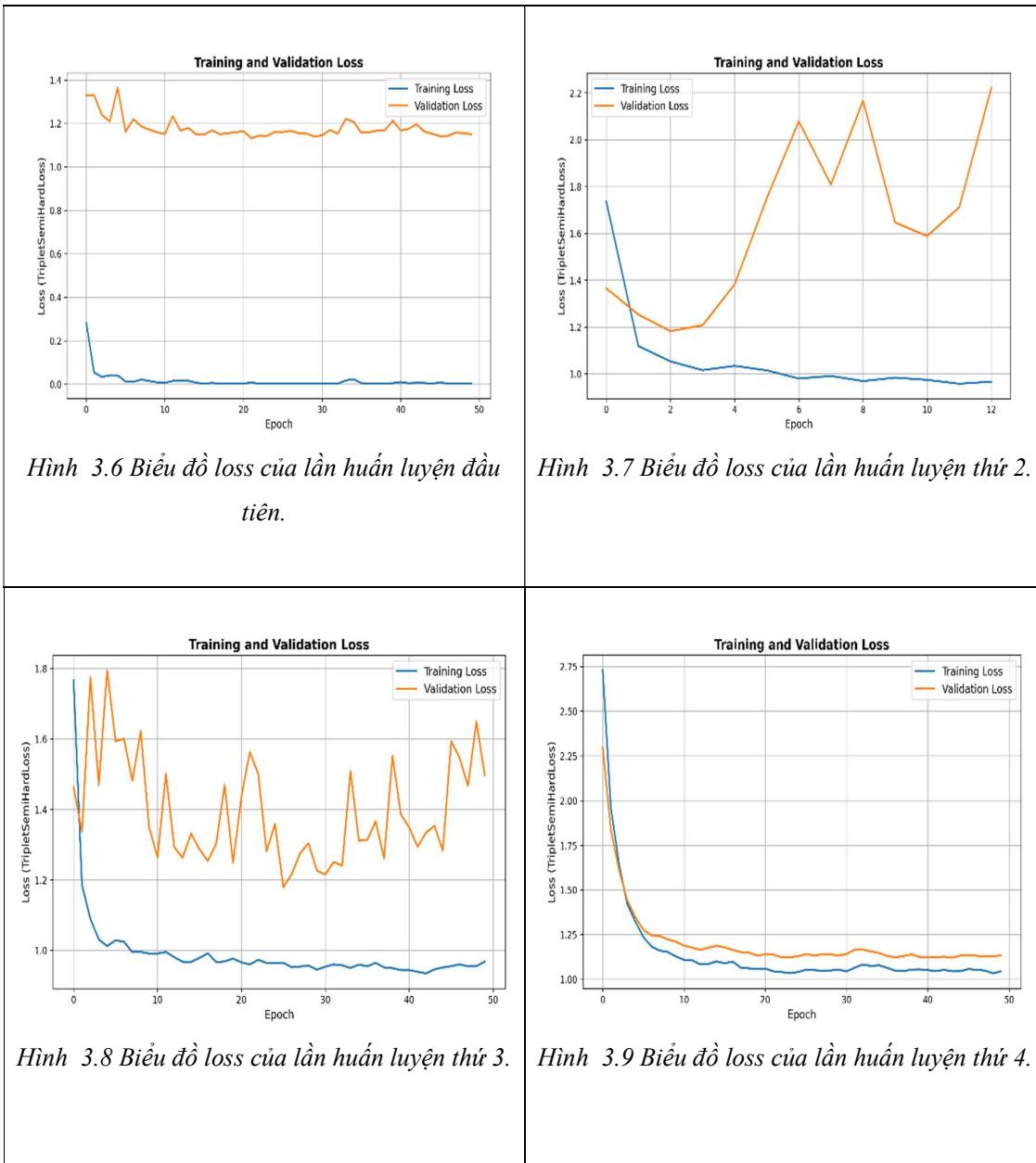
Để đánh giá hiệu quả của mô hình, các chỉ số như Accuracy, Precision, Recall và F1-score được sử dụng để đo lường khả năng dự đoán chính xác. Ngoài ra, ma trận nhầm lẫn (confusion matrix) cung cấp cái nhìn chi tiết về các lỗi dự đoán, giúp xác định mức độ nhầm lẫn giữa các sinh viên. Các biểu đồ trực quan, chẳng hạn như biểu đồ loss hoặc ma trận nhầm lẫn, hỗ trợ việc phân tích hiệu suất mô hình một cách trực quan.

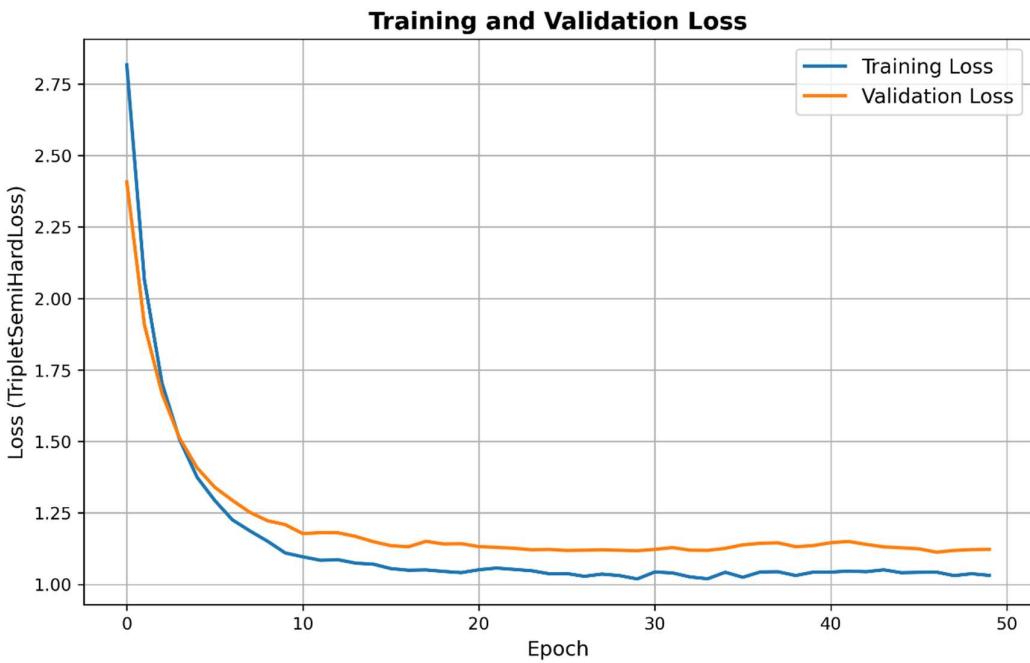
Các chỉ số đánh giá chính

- + **Accuracy (Độ chính xác):** đo lường tỷ lệ dự đoán đúng trên tổng số dự đoán.
Được tính theo công thức: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- + **Precision (Độ chính xác của dự đoán dương):** tỷ lệ các dự đoán dương đúng trong tổng các dự đoán là dương. Nó đo lường mức “tinh khiết” của nhãn dương do mô hình gán. Công thức tính: $Precision = \frac{TP}{TP+F}$
- + **Recall (Độ thu hồi):** tỷ lệ các mẫu dương thực sự được mô hình dự đoán đúng. Nó cho biết khả năng “phát hiện” hết các mẫu dương (trong các bài toán nhạy cảm, thiếu sót dương tính thường có chi phí cao). Công thức: $Recall = \frac{TP}{TP+FN}$
- + **F1-Score (Điểm F1):** trung bình điều hòa giữa Precision và Recall, dùng để cân bằng hai tiêu chí này. Khi độ lệch giữa Precision và Recall lớn, F1 sẽ ở gần giá trị nhỏ hơn trong hai giá trị đó. Công thức: $F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$
- + **Ma trận nhầm lẫn:** Hiển thị số lượng dự đoán đúng và sai cho từng danh tính, giúp phát hiện các trường hợp nhầm lẫn cụ thể.

Trong đó, TP (True Positive) và TN (True Negative) lần lượt là số mẫu dương và âm được mô hình phân loại đúng, còn FP (False Positive) và FN (False Negative) là số mẫu bị phân loại sai.

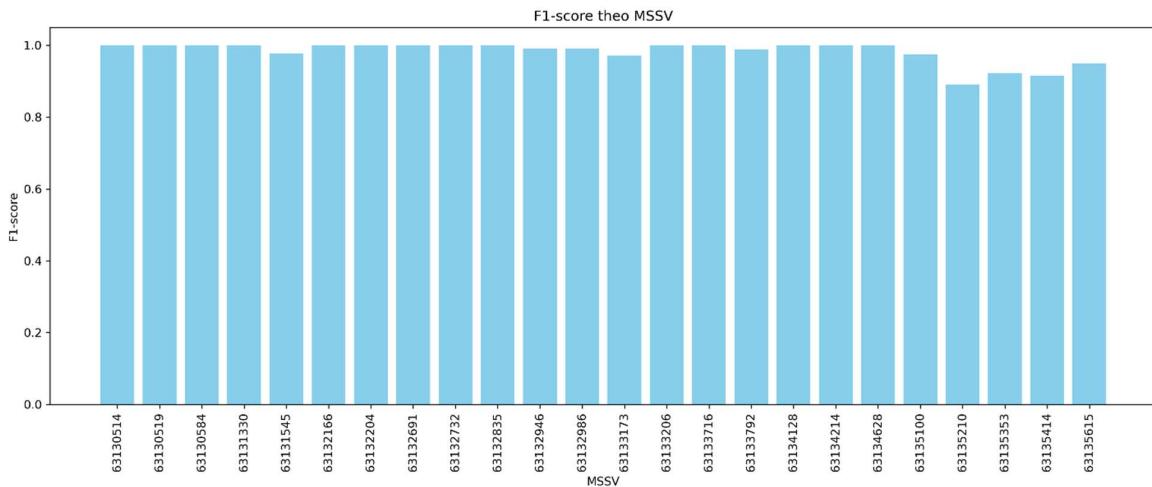
Biểu đồ ma trận nhầm lẫn hoặc biểu đồ loss được sử dụng để trực quan hóa hiệu suất mô hình, giúp dễ dàng nhận diện các vấn đề như nhầm lẫn danh tính hoặc overfitting.





Hình 3.10 Biểu đồ loss của lần huấn luyện thứ 5.

Qua 5 lần huấn luyện, với lần huấn luyện thứ 4 và thứ 5, khoảng cách giữa Training loss và Validation loss trong 2 biểu đồ là thấp nhất, chứng tỏ mô hình đã được cải thiện hơn. Dưới đây là ma trận nhầm lẫn (confusion matrix) và F1-score của mô hình huấn luyện thứ 5:

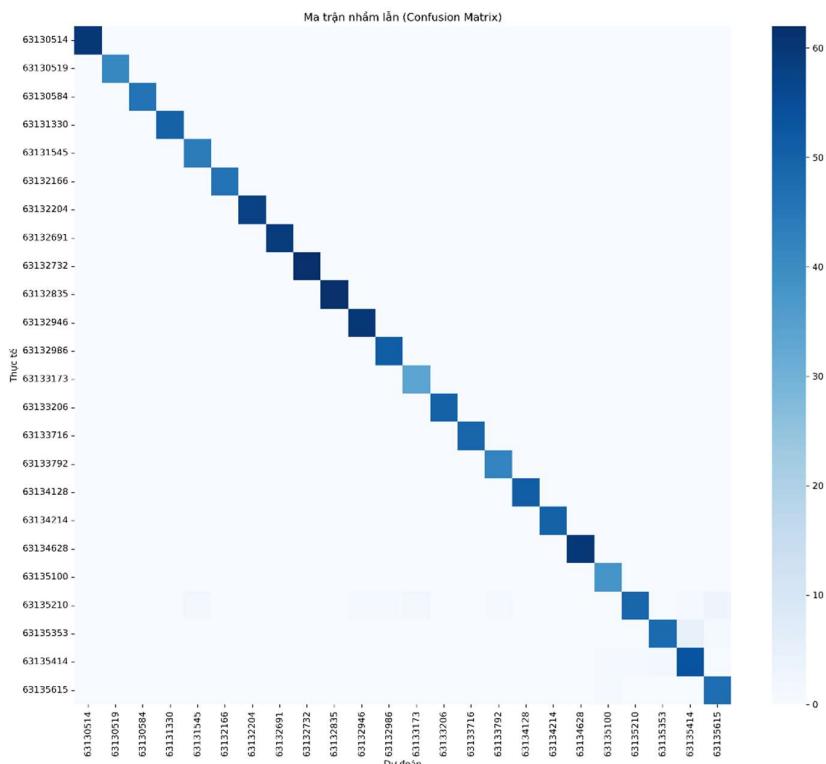


Hình 3.11 F1-score.

Biểu đồ “F1-score” là một biểu đồ cột, hiển thị điểm F1 cho 24 lớp, mỗi lớp được gắn nhãn bằng mã số sinh viên (MSSV). Trục hoành (x-axis) liệt kê các lớp, trong khi trục tung (y-axis) thể hiện điểm F1, dao động từ 0 đến 1, với các bước 0.2.

- + **Số lượng lớp:** Có 24 lớp, mỗi lớp đại diện cho một sinh viên, với nhãn như “63131330”, v.v.

- + **Phạm vi điểm F1:** Điểm F1 dao động từ khoảng 0.9 đến 1.0, với phần lớn các lớp đạt gần mức tối đa 1.0, cho thấy mô hình có hiệu suất cao trong việc nhận diện hầu hết sinh viên.
- + **Các lớp có điểm F1 thấp hơn:** Một số lớp như “63135353”, “63135414” có điểm F1 hơi thấp, nhưng vẫn trên 0,9. Điều này có thể do dữ liệu của các lớp này ít hơn hoặc có đặc điểm khuôn mặt khó phân biệt, cần thu thập thêm dữ liệu hoặc tinh chỉnh mô hình.
- + Biểu đồ cho thấy mô hình hoạt động rất tốt, với điểm F1 cao trên hầu hết các lớp, nhưng cần chú ý đến các lớp có điểm thấp hơn để cải thiện, có thể bằng cách thu thập thêm dữ liệu hoặc tinh chỉnh mô hình.



Hình 3.12 Ma trận nhầm lẫn (Confusion matrix)

Biểu đồ “Ma trận nhầm lẫn (Confusion Matrix)” là một ma trận vuông 24x24, với trục y là “Thực tế” (nhận thực tế) và trục x là “Dự đoán” (nhận dự đoán). Mỗi ô trong ma trận được tô màu, với thanh màu bên phải thể hiện thang giá trị từ 0 đến 60, màu đậm (xanh đậm) cho giá trị cao, màu nhạt (xanh nhạt hoặc trắng) cho giá trị thấp.

- + **Số lượng lớp:** Ma trận có 24 lớp, khớp với số lớp trong biểu đồ F1-score, với nhãn là các mã số sinh viên.

- + **Mẫu hình đường chéo:** Đường chéo chính (từ ô trên cùng bên trái đến ô dưới cùng bên phải) có màu đậm nhất, với giá trị lên đến 60, cho thấy mô hình dự đoán đúng nhiều trường hợp, thể hiện hiệu suất cao.
- + **Các ô ngoài đường chéo:** Có một số ô ngoài đường chéo có màu nhạt (giá trị 10-20), cho thấy nhầm lẫn giữa các lớp. Đặc biệt, lớp “63135353” và “63135210” có một số ô lân cận sáng hơn, cho thấy nhầm lẫn với các lớp gần, có thể do đặc điểm khuôn mặt tương tự.
- + **Mẫu hình tổng thể:** Ma trận có xu hướng đối xứng, với các ô xa đường chéo chính có màu nhạt (giá trị gần 0), cho thấy nhầm lẫn giữa các lớp xa nhau hiếm xảy ra.

Ma trận nhầm lẫn xác nhận mô hình có hiệu suất tốt, nhưng nhầm lẫn giữa các lớp gần nhau (như “63135353”, “63135210” với các lớp lân cận) cần được cải thiện, có thể do dữ liệu không đủ đa dạng hoặc đặc điểm khuôn mặt tương đồng.

3.3.4 Chuyển đổi sang TfLite (TensorFlow Lite)

Phần này mô tả quá trình chuyển đổi mô hình FaceNet từ định dạng Keras (HDF5) sang TensorFlow Lite (TFLite) để triển khai trên thiết bị Android. Mục tiêu là tạo ra mô hình nhẹ, tối ưu về kích thước và hiệu suất, phù hợp cho ứng dụng điểm danh sinh viên bằng nhận diện khuôn mặt.

Quy trình chuyển đổi

Chuẩn bị môi trường:

Quá trình chuyển đổi sử dụng các thư viện tensorflow và tensorflow_addons để xử lý mô hình Keras. Thư viện tensorflow cung cấp các công cụ chính cho việc tải và chuyển đổi mô hình, trong khi tensorflow-addons hỗ trợ hàm măt măt tùy chỉnh. Logging được cấu hình với mức độ INFO để ghi lại thông tin và ERROR để theo dõi lỗi, đảm bảo dễ dàng phát hiện vấn đề trong quá trình thực hiện.

Tải mô hình Keras:

Mô hình FaceNet, được lưu trong file face_auth_model_facenet.h5, được tải bằng hàm tf.keras.models.load_model. Do mô hình sử dụng hàm măt măt tùy chỉnh TripletSemiHardLoss, tham số custom_objects được khai báo để ánh xạ hàm này, đảm bảo mô hình được tải chính xác mà không gặp lỗi cấu hình.

Chuyển đổi sang TFLite:

Quá trình chuyển đổi bắt đầu bằng việc tạo một đối tượng TFLiteConverter từ mô hình Keras thông qua `tf.lite.TFLiteConverter.from_keras_model`. Tùy chọn tối ưu hóa quantization (`tf.lite.Optimize.DEFAULT`) được áp dụng để giảm kích thước mô hình và tăng tốc độ xử lý bằng cách chuyển các trọng số sang kiểu dữ liệu nhỏ hơn (như `int8` hoặc `float16`). Sau khi chuyển đổi, mô hình được lưu vào file `face_auth_model_facenet.tflite` dưới dạng nhị phân.

Toàn bộ quy trình được bao bọc trong khái try-except để xử lý các lỗi tiềm ẩn, như file mô hình không tồn tại hoặc lỗi cấu hình. Các lỗi được ghi lại vào log với thông tin chi tiết, hỗ trợ việc gỡ lỗi và đảm bảo quá trình chuyển đổi diễn ra suôn sẻ.

3.3.5 So sánh giữa file H5 và TFLite

Bảng 3.1 Sự khác nhau giữa H5 và TFLite

Tiêu chí	File H5	File TFLite
Định dạng	HDF5, lưu trữ mô hình Keras với toàn bộ kiến trúc, trọng số và siêu dữ liệu.	TFLite, định dạng nhẹ của TensorFlow cho thiết bị di động và nhúng.
Kích thước	Lớn hơn (thường vài chục MB), do chứa thông tin đầy đủ của mô hình.	Nhỏ hơn đáng kể, nhờ quantization, phù hợp với bộ nhớ hạn chế của Android.
Hiệu suất	Phù hợp cho máy tính với tài nguyên mạnh, xử lý chậm hơn trên thiết bị di động.	Tối ưu cho thiết bị di động, xử lý nhanh hơn, hỗ trợ nhận diện thời gian thực.
Tích hợp	Khó tích hợp trực tiếp vào ứng dụng Android, cần môi trường Python.	Dễ tích hợp vào Android qua TensorFlow Lite Interpreter (FaceRecognitionHelper).
Độ chính xác	Độ chính xác cao, sử dụng <code>float32</code> cho trọng số.	Có sự giảm nhẹ độ chính xác do quantization, nhưng vẫn đáp ứng yêu cầu, sử dụng <code>int8</code> làm trọng số.

3.4 XÂY DỰNG ỦNG DỤNG

Phần này trình bày chi tiết quá trình phát triển ứng dụng điểm danh sinh viên sử dụng công nghệ nhận diện khuôn mặt, được triển khai trên nền tảng Android với sự hỗ trợ của các công cụ như Google ML Kit, TensorFlow Lite, OpenCV, và Jetpack Compose. Ứng dụng được thiết kế để tự động hóa quy trình điểm danh, đảm bảo tính chính xác, nhanh chóng và tiện lợi trong môi trường lớp học. Nội dung bao gồm các thành phần chính của ứng dụng, quy trình hoạt động, và cách tích hợp các công nghệ để đáp ứng các yêu cầu của bài toán.

3.4.1 Mô tả ứng dụng

Ứng dụng được xây dựng nhằm mục đích thay thế phương pháp điểm danh thủ công, cho phép giảng viên ghi nhận sự hiện diện của sinh viên thông qua nhận diện khuôn mặt từ camera của thiết bị di động. Ứng dụng hoạt động theo thời gian thực, sử dụng camera trước (ưu tiên) hoặc camera sau của thiết bị để chụp ảnh hoặc video, sau đó thực hiện các bước phát hiện khuôn mặt, trích xuất đặc trưng, và xác định danh tính sinh viên dựa trên cơ sở dữ liệu đã được huấn luyện trước. Kết quả điểm danh được lưu trữ dưới dạng file CSV, kèm theo thông tin về mã số sinh viên (MSSV), tên, và thời gian điểm danh.

Các tính năng chính của ứng dụng bao gồm:

- **Phát hiện khuôn mặt:** Sử dụng Google ML Kit Face Detection để xác định vị trí khuôn mặt trong khung hình.
- **Nhận diện danh tính:** Tích hợp mô hình FaceNet (định dạng TensorFlow Lite) để trích xuất vector đặc trưng (embedding) và so sánh với cơ sở dữ liệu để xác định MSSV và tên sinh viên.
- **Giao diện người dùng:** Được xây dựng bằng Jetpack Compose, hiển thị hình ảnh từ camera, vẽ hộp giới hạn (bounding box) quanh khuôn mặt, và hiển thị kết quả nhận diện (MSSV, tên, thời gian).
- **Lưu trữ kết quả:** Lưu thông tin điểm danh vào file CSV với định dạng “MSSV, Name, Time” và đảm bảo không ghi trùng lặp.
- **Quản lý tài nguyên:** Xóa các file ảnh tạm sau khi lưu kết quả vào CSV và thực hiện dọn dẹp khi thoát ứng dụng.

Ứng dụng được thiết kế để hoạt động hiệu quả trên các thiết bị Android, với giao diện thân thiện và quy trình sử dụng đơn giản, phù hợp cho mọi người sử dụng trong môi trường lớp học.

3.4.2 Thành phần chính của ứng dụng

Ứng dụng được xây dựng dựa trên ba thành phần chính, được triển khai thông qua các file mã nguồn chính: FaceRecognitionHelper.kt, MainActivity.kt, và MainScreen.kt. Dưới đây là mô tả chi tiết về vai trò của từng thành phần:

*FaceRecognitionHelper.kt

FaceRecognitionHelper.kt là lớp trung tâm chịu trách nhiệm xử lý logic nhận diện khuôn mặt và quản lý dữ liệu điểm danh. Lớp này tích hợp Google ML Kit Face Detection để phát hiện khuôn mặt, TensorFlow Lite để trích xuất đặc trưng, và các hàm phụ trợ để lưu trữ kết quả. Các chức năng chính bao gồm:

- **Khởi tạo mô hình và dữ liệu:**

- + Tải mô hình FaceNet từ file face_auth_model_facenet.tflite thông qua TensorFlow Lite Interpreter.
- + Đọc dữ liệu từ file embeddings.json (chứa các vector đặc trưng của sinh viên) và students.json (chứa ảnh xạ MSSV - Tên sinh viên).
- + Khởi tạo bộ phát hiện khuôn mặt của Google ML Kit với cấu hình ưu tiên tốc độ (PERFORMANCE_MODE_FAST) và ngưỡng kích thước khuôn mặt tối thiểu là 0.3.

- **Xử lý hình ảnh:**

- + Nhận ảnh đầu vào từ camera (dạng Bitmap), xoay ảnh.
- + Sử dụng Google ML Kit để phát hiện khuôn mặt, trả về tọa độ hộp giới hạn (bounding box) của khuôn mặt đầu tiên.
- + Cắt vùng khuôn mặt từ ảnh gốc, chuẩn hóa về kích thước 224x224 pixel để phù hợp với đầu vào của FaceNet.
- + Trích xuất vector đặc trưng (embedding) 128 chiều từ khuôn mặt đã cắt bằng mô hình FaceNet.
- + So sánh embedding với cơ sở dữ liệu trong embeddings.json bằng khoảng cách Cosine (cosineDistance), với ngưỡng 1.5 để xác định danh tính (nếu khoảng cách nhỏ hơn ngưỡng, trả về MSSV tương ứng; ngược lại trả về “Unknown”).

- **Lưu trữ kết quả:**
 - + Lưu ảnh chụp khuôn mặt vào bộ nhớ thiết bị với tên định dạng capture_[timestamp].jpg.
 - + Ghi kết quả điểm danh (MSSV, tên, thời gian) vào file CSV (Attendance_[date].csv), đảm bảo không ghi trùng lặp thông qua kiểm tra MSSV đã tồn tại.
 - + Xóa ảnh tạm sau khi lưu vào CSV để tiết kiệm bộ nhớ.
- **Dọn dẹp tài nguyên:**
 - + Xóa tất cả ảnh tạm (định dạng .jpg) khi thoát ứng dụng.
 - + Loại bỏ các bản ghi trùng lặp trong file CSV, đảm bảo mỗi MSSV chỉ xuất hiện một lần.

*MainActivity.kt

MainActivity.kt là lớp chính điều phối hoạt động của ứng dụng, quản lý camera, giao tiếp với FaceRecognitionHelper, và hiển thị giao diện người dùng thông qua Jetpack Compose. Các chức năng chính bao gồm:

- **Quản lý quyền camera:**
 - + Kiểm tra và yêu cầu quyền truy cập camera (Manifest.permission.CAMERA) khi ứng dụng khởi động.
 - + Nếu quyền được cấp, khởi động camera; nếu không, hiển thị thông báo và thoát ứng dụng.
- **Khởi động camera:**
 - + Sử dụng Android CameraX để thiết lập camera (ưu tiên camera trước, nếu không có thì dùng camera sau).
 - + Cấu hình độ phân giải camera là 640x480 pixel, sử dụng Preview để hiển thị hình ảnh từ camera và ImageAnalysis để phân tích khung hình.
 - + Phân tích khung hình mỗi 500ms để tránh quá tải, chuyển khung hình thành Bitmap và gửi đến FaceRecognitionHelper để xử lý.
- **Xử lý kết quả nhận diện:**
 - + Nhận kết quả từ FaceRecognitionHelper (chuỗi thông báo, tọa độ hộp giới hạn, và đối tượng RecognitionResult nếu nhận diện thành công).
 - + Cập nhật trạng thái giao diện (kết quả nhận diện, hộp giới hạn, và thông tin sinh viên) thông qua các biến trạng thái (mutableStateOf).

- + Tạm dừng camera khi nhận diện thành công để hiển thị ảnh chụp và thông tin sinh viên.

- **Quản lý vòng đời ứng dụng:**

- + Khởi tạo FaceRecognitionHelper khi ứng dụng bắt đầu.
- + Dọn dẹp tài nguyên (tắt camera, hủy coroutine) khi ứng dụng bị hủy.

***MainScreen.kt**

MainScreen.kt chịu trách nhiệm xây dựng giao diện người dùng bằng Jetpack Compose, cung cấp trải nghiệm trực quan và thân thiện. Các thành phần giao diện bao gồm:

- **Khu vực hiển thị camera:**

- + Sử dụng AndroidView để nhúng PreviewView của CameraX, hiển thị hình ảnh trực tiếp từ camera.
- + Nếu nhận diện thành công, ẩn PreviewView và hiển thị ảnh chụp được lưu từ RecognitionResult.capturedImage bằng Image của Coil.

- **Hộp giới hạn khuôn mặt:**

- + Vẽ hộp giới hạn màu xanh quanh khuôn mặt trên PreviewView hoặc ảnh chụp bằng Canvas, với tọa độ được điều chỉnh tỷ lệ dựa trên kích thước màn hình.

- **Hiển thị kết quả nhận diện:**

- + Nếu chưa nhận diện, hiển thị dòng chữ “Đang chờ nhận diện...” bằng Text.
- + Nếu nhận diện thành công, hiển thị thông tin MSSV, tên, và thời gian từ RecognitionResult.

- **Nút điều khiển:**

- + Nút “Save”: Lưu kết quả điểm danh vào file CSV, chỉ kích hoạt khi nhận diện thành công.
- + Nút “Chụp lại”: Khởi động lại camera để tiếp tục nhận diện.
- + Nút “Thoát”: Hiển thị hộp thoại xác nhận thoát, khi xác nhận sẽ gọi FaceRecognitionHelper.cleanupOnExit() và thoát ứng dụng.

3.5 QUY TRÌNH SỬ DỤNG CÔNG CỤ

3.5.1 Quy trình xử lý dữ liệu

- **Thu thập dữ liệu:**
 - + Chụp ảnh khuôn mặt sinh viên (khoảng 20 ảnh/người) từ nhiều góc độ và điều kiện ánh sáng.
 - + Lưu ảnh vào thư mục dataset/faces-63cont-clc, phân loại theo MSSV (ví dụ: 63131330, 63133716).
- **Tiền xử lý dữ liệu:**
 - + Sử dụng script preprocess_image.py để xử lý ảnh:
 - + Đọc ảnh bằng OpenCV từ thư mục đầu vào.
 - + Chuyển ảnh sang grayscale và sử dụng Haar Cascade (haarcascade_frontalface_alt2.xml) để phát hiện khuôn mặt với scaleFactor=1.1, minNeighbors=5, minSize=(30, 30).
 - + Cắt vùng khuôn mặt đầu tiên, resize về 224x224 pixel, chuyển từ BGR sang RGB, chuẩn hóa giá trị pixel (chia cho 255, đưa về [0, 1]).
 - + Chuyển ảnh sang định dạng PIL và lưu dưới dạng jpg (chất lượng 95%) vào thư mục processed_data, phân loại theo MSSV.
- **Tạo thông tin sinh viên:**
 - + Tạo file students.txt chứa thông tin sinh viên với định dạng MSSV [khoảng trắng] Tên (ví dụ: 63131330 Vu Minh Thinh).

3.5.2 Quy trình xây dựng mô hình

Quy trình xây dựng mô hình FaceNet được thực hiện thông qua script train_facenet.py và generate_embeddings.py để tạo mô hình và embedding vector cho sinh viên.

- **Tải dữ liệu:**
 - + Sử dụng hàm load_data trong train_facenet.py để:
 - Tải ảnh từ processed_data, đọc bằng load_img với kích thước 224x224, chuẩn hóa pixel (chia 255).
 - Gán nhãn số nguyên cho mỗi MSSV, tạo mảng images, labels, và danh sách mssv_list.
- **Xây dựng mô hình FaceNet:**
 - + Tạo mô hình tuần tự trong create_facenet_model với:

- 3 tầng Conv2D (64, 128, 256 filters, kernel 3x3, ReLU) xen kẽ MaxPooling2D (2x2).
 - Tầng Dropout (0.5) và Dense (128 chiều) với chuẩn hóa L2 (l2_normalize) để tạo embedding 128 chiều.
- + Biên dịch mô hình với Adam (learning rate 0.0001) và hàm mất mát TripletSemiHardLoss từ TensorFlow Addons.
- **Tăng cường dữ liệu:**
 - + Áp dụng ImageDataGenerator với: xoay 20°, dịch chuyển 5%, lật ngang, tăng độ sáng 6%, zoom ±2%, fill constant 0.9.
- **Huấn luyện mô hình:**
 - + Chia dữ liệu: 80% huấn luyện, 20% kiểm tra.
 - + Huấn luyện với 50 epoch, batch size 32, sử dụng datagen.flow cho tập huấn luyện và dữ liệu gốc cho tập kiểm tra.
 - + Lưu mô hình vào face_auth_model_facenet.h5 và lịch sử huấn luyện vào history.pkl.
- **Tạo embedding:**
 - + Sử dụng generate_embeddings.py để:
 - Tải mô hình face_auth_model_facenet.h5.
 - Đọc ảnh từ processed_data, resize về 224x224, chuẩn hóa pixel, đưa qua mô hình để tạo embedding 128 chiều.
 - Lưu embedding và MSSV vào embeddings.pkl với cấu trúc {'embeddings': [(embedding, mssv)], 'mssv_list': [mssv]}.
- **Kết quả:**
 - + Mô hình FaceNet tạo embedding chính xác, lưu trữ trong face_auth_model_facenet.h5 và embeddings.pkl.

3.5.3 Quy trình chuyển đổi

Quy trình chuyển đổi mô hình và dữ liệu sang định dạng phù hợp cho ứng dụng Android được thực hiện thông qua convert_to_tflite.py và convert_data_to_json.py.

- **Chuyển đổi mô hình sang TFLite:**
 - + Sử dụng convert_to_tflite.py:
 - Tải mô hình Keras (face_auth_model_facenet.h5) với custom_objects cho TripletSemiHardLoss.

- Tạo TFLiteConverter, áp dụng quantization (tf.lite.Optimize.DEFAULT) để sử dụng int8 cho trọng số, giảm kích thước và tăng tốc độ.
 - Chuyển đổi và lưu mô hình vào face_auth_model_facenet.tflite.
- **Chuyển đổi dữ liệu sang JSON:**
 - + Sử dụng convert_data_to_json.py:
 - Tải embeddings.pkl, chuyển embedding thành danh sách và lưu vào embeddings.json với cấu trúc {'array': [{'mssv': mssv, 'embedding': [values]}]}.
 - Đọc students.txt, tạo dictionary {'mssv': 'name'} và lưu vào students.json.
- **Kết quả:**
 - + File face_auth_model_facenet.tflite có kích thước nhỏ gọn, phù hợp cho Android.
 - + File embeddings.json và students.json sẵn sàng tích hợp vào thư mục assets của ứng dụng.

3.5.4 Tích hợp trên thiết bị di động

- **Chuẩn bị tài nguyên:**
 - + Sao chép file face_auth_model_facenet.tflite, embeddings.json, và students.json vào thư mục assets của dự án Android.
- **Phát triển ứng dụng:**
 - + Sử dụng Android Studio với Jetpack Compose, CameraX, Google ML Kit, và TensorFlow Lite để xây dựng ứng dụng.
 - + Tích hợp logic nhận diện trong FaceRecognitionHelper.kt:
 - Tải mô hình TFLite và dữ liệu JSON.
 - Phát hiện khuôn mặt bằng Google ML Kit, trích xuất embedding bằng TFLite, và so sánh với embeddings.json.
 - + Xây dựng giao diện trong MainScreen.kt để hiển thị camera, hộp giới hạn, và kết quả nhận diện.
 - + Quản lý camera và trạng thái trong MainActivity.kt.

– **Triển khai ứng dụng:**

- + Biên dịch và cài đặt ứng dụng lên thiết bị Android, đảm bảo quyền truy cập camera được cấp.

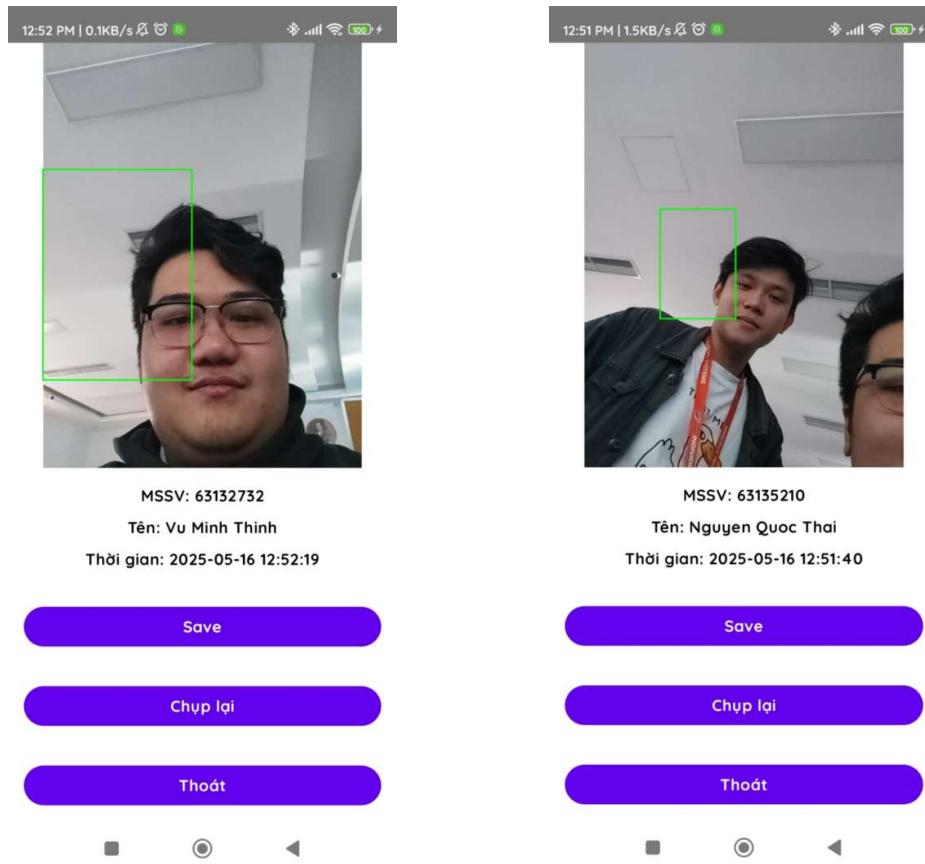
3.5.5 Sử dụng trên thiết bị di động

– **Khởi động ứng dụng:**

- + Mở ứng dụng, cấp quyền camera. Camera trước (hoặc sau) khởi động, hiển thị hình ảnh trực tiếp.

– **Nhận diện khuôn mặt:**

- + Hướng camera vào khuôn mặt sinh viên. Google ML Kit phát hiện khuôn mặt, vẽ hộp giới hạn màu xanh.
- + Mô hình TFLite trích xuất embedding, so sánh với embeddings.json để xác định MSSV và tên (ngưỡng Cosine < 1.5).



Hình 3.13 Giao diện điểm danh của sinh viên

– **Lưu kết quả:**

- + Nếu nhận diện thành công, camera tạm dừng, hiển thị ảnh chụp, MSSV, tên, và thời gian.

- + Nhấn “Save” để lưu vào file CSV (Attendance_[date].csv), ảnh tạm được xóa.
- + Nhấn “Chụp lại” để tiếp tục điểm danh.

	A	B	C	D	E	F
1	MSSV,Name,Time					
2	63132732,Vu Minh Thinh,2025-05-16 12:48:21					
3	63135210,Nguyen Quoc Thai,2025-05-16 12:48:41					
4						
5						
6						
7						
8						

Hình 3.14 Giao diện xuất file CSV

- **Thoát ứng dụng:**
 - + Nhấn “Thoát”, xác nhận qua hộp thoại. Ứng dụng xóa ảnh tạm, loại bỏ bản ghi trùng lặp trong CSV, rồi đóng.

Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 KẾT LUẬN

Trong khuôn khổ đề tài này, tôi đã thành công trong việc ứng dụng học sâu và mô hình FaceNet để xây dựng một ứng dụng nhận diện khuôn mặt phục vụ điểm danh sinh viên trên nền tảng Android. Cụ thể, tôi đã sử dụng Google ML Kit để phát hiện khuôn mặt từ luồng camera, kết hợp với mô hình FaceNet để trích xuất vector đặc trưng 128 chiều, từ đó xác định danh tính sinh viên. Bộ dữ liệu ảnh khuôn mặt đã được thu thập, tiền xử lý bằng Haar Cascade, và chuẩn hóa thành định dạng phù hợp để huấn luyện mô hình. Môi trường phát triển ứng dụng được thiết lập với Jetpack Compose, CameraX, và TensorFlow Lite, cho phép nhận diện thời gian thực với độ chính xác cao. Ứng dụng được phát triển có khả năng tự động hóa quy trình điểm danh, hiển thị thông tin sinh viên (MSSV, tên, thời gian) kèm hộp giới hạn quanh khuôn mặt, và lưu kết quả vào file CSV. Giao diện thân thiện, dễ sử dụng, hỗ trợ người dùng quản lý điểm danh hiệu quả, giảm thiểu thời gian và công sức so với phương pháp thủ công. Tuy nhiên, hệ thống vẫn cần cải thiện để đáp ứng tốt hơn trong các điều kiện thực tế, đặc biệt khi xử lý dữ liệu lớn hoặc hoạt động trong môi trường ánh sáng yếu.

4.2 HƯỚNG PHÁT TRIỂN

Trong tương lai, việc nâng cấp hệ thống lưu trữ dữ liệu là một ưu tiên quan trọng để tăng tính linh hoạt và hiệu quả của ứng dụng. Thay vì sử dụng file text (students.txt) và JSON, việc tích hợp cơ sở dữ liệu như SQLite hoặc Firebase sẽ cho phép quản lý thông tin sinh viên một cách khoa học, hỗ trợ cập nhật và truy xuất dữ liệu nhanh chóng, đặc biệt khi số lượng sinh viên tăng lên. Đồng thời, phát triển một giao diện quản lý trực tiếp trên ứng dụng hoặc kết nối với hệ thống quản lý học tập của trường sẽ giúp tự động hóa nhập liệu, nâng cao bảo mật và giảm thiểu lỗi nhập tay.

Một hướng cải tiến khác là tối ưu hóa việc hiển thị hộp giới hạn quanh khuôn mặt để đảm bảo độ chính xác và đồng bộ với vị trí khuôn mặt. Hiện tại, sự không khớp về tỷ lệ giữa khung hình camera và màn hình thiết bị có thể được giải quyết bằng cách tự động điều chỉnh tỷ lệ dựa trên kích thước thực tế, hoặc áp dụng mô hình phát hiện khuôn mặt tiên tiến như MTCNN để tăng độ chính xác, đặc biệt trong các điều kiện ánh sáng phức tạp.

Để nâng cao hiệu suất tổng thể, ứng dụng cần cải thiện khả năng nhận diện trong môi trường ánh sáng yếu hoặc khi có nhiều người trong khung hình. Thu thập dữ liệu

huấn luyện đa dạng hơn, kết hợp với các kỹ thuật tiền xử lý nâng cao như histogram equalization, sẽ giúp mô hình hoạt động ổn định hơn. Ngoài ra, tối ưu hóa mô hình FaceNet bằng các kỹ thuật như quantization-aware training hoặc pruning sẽ giảm tài nguyên tính toán, cải thiện tốc độ xử lý trên các thiết bị Android cấu hình thấp.

Hơn nữa, việc bổ sung các tính năng mới sẽ tăng cường tính ứng dụng của hệ thống. Tích hợp xác thực đa yếu tố, chẳng hạn như mã QR hoặc mã PIN, sẽ nâng cao độ tin cậy của quá trình điểm danh. Phát triển tính năng xuất báo cáo điểm danh dưới dạng Excel hoặc thống kê theo thời gian sẽ hỗ trợ giảng viên quản lý hiệu quả hơn. Những cải tiến này sẽ giúp ứng dụng trở nên mạnh mẽ, phù hợp hơn với nhu cầu thực tế, và mở rộng tiềm năng ứng dụng trong giáo dục.

Chương 5. TÀI LIỆU THAM KHẢO

- [1] “Rapid Object Detection using a Boosted Cascade of Simple Features,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.
- [2] “OpenFace: A General-Purpose Face Recognition Library with Mobile Applications,” 2016.
- [3] “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, 1998.
- [4] “FaceNet: A Unified Embedding for Face Recognition and Clustering,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [5] “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [6] “Deep Residual Learning for Image Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [7] OpenCV Team, “Open Source Computer Vision Library,” 2000. [Online]. Available: <https://opencv.org>. [Accessed 05 06 2025].
- [8] Chinese University of HongKong, “CelebFaces Attributes Dataset (CelebA),” 2015. [Online]. Available: <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>. [Accessed 05 06 2025].
- [9] “Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments,” 2007. [Online]. Available: <http://vis-www.cs.umass.edu/lfw/>. [Accessed 05 06 2025].
- [10] ImageNet Project, “ImageNet: A Large-Scale Hierarchical Image Database,” 2009. [Online]. Available: <https://www.image-net.org>. [Accessed 05 06 2025].